Software Engineering Institute

The Personal Software Process[SM] (PSP[SM]) Body of Knowledge, Version 1.0

Marsha Pomeroy-Huff
Julia Mullaney
Robert Cannon
Mark Seburn

Foreword by Watts S. Humphrey

August 2005

Since its initial release, this report has been revised.
This revision was released March 26, 2008.

SPECIAL REPORT
CMU/SEI-2005-SR-003

Software Engineering Process Management Program

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

About This Report

The intent of the Personal Software Process[SM] (PSP[SM]) body of knowledge (BOK) contained in this report is to provide guidance to software professionals who are interested in using proven-effective, disciplined methods to improve their personal software development process. However, it is also of interest to individuals who do not develop software but work with or manage projects involving many other kinds of development. The PSP BOK can aid these individuals in determining the knowledge and skills that most development professionals should possess. Development professionals who will find the PSP BOK to be useful include but are not limited to

- senior executives of software development organizations or of companies who use software as a component in their products
- program and project managers
- members of integrated product-development teams
- professionals who give support to software and other development projects (for example, testers, quality assurance specialists, technical writers)
- customers and stakeholders
- process improvement consultants

The PSP BOK can also be used by education professionals in creating or assessing instructional products, from individual courses to entire curricula. For example, it can be used by professors or course designers to ensure that the content of new courses enables students to master the knowledge and skills of each competency area or to examine existing courses and evaluate them on their coverage of the requisite competencies.

Similarly, this document can be used to create, assess, or accredit certifications or other credentials programs for PSP practitioners. Certification programs provide individuals with documentation attesting that those individuals have attained a well-defined and objectively-measured level of proficiency in a particular field or discipline, as defined by a core set of knowledge and skills. Individuals who successfully demonstrate their proficiency—usually measured by performance on a standardized assessment instrument, and recognized by awarding of a credential or certification—are regarded as competent, skilled professionals with a demonstrated level of mastery in the competencies delineated by their profession's body of knowledge.

[SM] Personal Software Process and PSP are service marks of Carnegie Mellon University.

Acknowledgements

In preparing this report, the authors consulted with several individuals who provided ideas and contributions to the content of the PSP BOK. In particular, we want to acknowledge Jefferson Welch and Alan Willet for helping us out when we got stuck for inspiration, words, and/or time (sometimes all at the same moment). Our editors, Susan Kushner and Clare Dixon, were instrumental in catching potentially embarrassing typos, grammar errors, and other defects that eluded us. They also guided the report through the various review and approval steps in record time. Other individuals also contributed to reviewing the content and clarity of the report, and we would like to thank these individuals for their time and assistance: Yoshi Akiyama, Kimberly Campbell, David Carrington, Noopur Davis, Caroline Graettinger, Tom Hilburn, Watts Humphrey, Jim Over, Hans-Peter Pfister, Mary Ellen Rich, Jim Van Buren, and Alan Willet.

Finally, the authors would like to honor the ancient tradition of "saving the best for last" by ending this section in acknowledging the enormous contributions of Watts Humphrey, not only to this team and to this effort, but to the entire field of software engineering. It is fitting that as Version 1.0 of this document was being completed, Watts received word that he had been awarded the National Medal of Technology, which is given to America's leading innovators by the President of the United States. This award is appropriate recognition for a lifetime of effort aimed at improving the software development process. The Capability Maturity Model® (CMM®) has helped many companies to achieve consistent excellence at the organizational level, and the culmination of Watts's life work to date—the PSP and TSP methodologies—have given that same capability to individuals and teams. Those of us who use these methods thank Watts for these innovations, which have improved not only the quality of our work and our schedule- and effort-estimating abilities, but also the quality of our work life in general.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Foreword

What do we want from software engineering? That, of course, depends on who is asking. Just about everyone wants quality software on predictable schedules and for committed costs. Those of us in the development business also want a fun job, a rewarding development experience, and the satisfaction of doing professional work. Managers and users want development professionals who are credible and professional and can be trusted to do what they commit to do. Educators are most concerned with preparing student developers to do the kinds of work that society needs.

While the software business has been troubled almost from the outset, with unpredictable schedules, missed commitments, and poor-quality products, it has also produced some remarkable innovations. Software has supported and enabled many, if not most, of the modern advances in science and technology. Software is a truly extraordinary technology. It controls, guides, or enables just about every product, tool, or support system in our modern world. Just about everything that people now do depends to some degree on the effective performance of software. Software is central to our lives, to our businesses, and, in a growing number of cases, to our very survival. It is therefore critically important that software professionals learn and consistently use the best possible methods when they do their jobs.

This body of knowledge encapsulates the basic knowledge and principles behind the PSP. While the PSP is almost certainly not the last word in software development practice, it was developed from the basic principles of science and engineering, and when developers truly follow these principles in their work, they produce quality products on predictable schedules and for their committed costs. This may sound too good to be true, but it is not really the PSP that does it. The PSP is so effective because software professionals are such extraordinarily capable people. Since the PSP is a set of practices and methods that enable software developers to control their own working lives, when competent professionals learn and consistently follow these engineering and scientific principles, and when they are empowered to manage their own work, they do an unbelievably good job.

This BOK describes the basic PSP practices and methods. From this foundation, it is our hope that the software community will develop the courses, the support tools and methods, the certification and qualification programs, and all of the other required elements to enable the widespread adoption of these methods.

Watts S. Humphrey
July 2005

Abstract

As the profession of software engineering evolves and matures, it must achieve some of the critical elements needed for recognition as a bona fide discipline. Among these elements are the establishment of a recognized body of knowledge (BOK) and certification of professional practitioners.

The body of knowledge contained in this report is designed to complement the IEEE Computer Society's *Software Engineering Body of Knowledge (SWEBOK)* by delineating the key skills and concepts that compose the knowledge areas and competencies of a proven-effective process improvement method, the Personal Software Process (PSP). As adoption of the PSP methodology continues to grow, it becomes crucial to document the fundamental knowledge and skills that set PSP practitioners apart from other software engineers. The PSP BOK serves this purpose and more. It helps individual practitioners to assess and improve their own skills; provides employers with an objective baseline for assessing the personal process skills and capabilities of their engineers and product development teams; and guides academic institutions that want to incorporate PSP into their software and other engineering courses or curricula. The PSP BOK also facilitates the development of PSP certification programs that are based on a well-established, standard set of knowledge and skills.

1 Introduction

Software engineering is one of the largest and most influential industries in modern society. It has evolved from early calculation applications used only by government agencies and university think-tanks to complex applications that permeate every aspect of modern life. The banking, telecommunications, travel, medical, entertainment, and even agriculture industries rely heavily on software to operate. Software affects even the most mundane aspects of our lives, from buying groceries to doing a load of laundry or filling our cars' fuel tanks with gas.

Yet, in spite of its pervasive influence, software engineering is a relatively young discipline. The term "software engineering" has been in popular use for less than 40 years, following its introduction in the title of a NATO Science Committee conference at Garmisch, Germany [Naur 69].

One frequent criticism of the software engineering profession is the poor quality of the products it produces. This problem has been attributed to many causes, from the way software engineers are educated to the overall problems inherent within a young profession. An article in the online encyclopedia *Wikipedia* summed up the criticism of software engineering as follows:

> *"In traditional engineering, there is a clear consensus how things should be built, which standards should be followed, and which risks must be taken care of; if an engineer does not follow these practices and something fails, he gets sued. There is no such consensus in software engineering: Everyone promotes their own methods, claiming huge benefits in productivity, usually not backed up by any scientific, unbiased evidence"* [Wikipedia 05].

A powerful counter to this criticism is the widespread adoption of the Personal Software Process (PSP) methodology. Developed in 1993 by Watts S. Humphrey, the PSP is a disciplined and structured approach to developing software. By using the PSP concepts and methods in their work, engineers in almost any technical field can improve their estimating and planning skills, make commitments that they can meet, manage the quality of their work, and reduce the number of defects in their products.

The effectiveness of the PSP methodology (and its companion technology, the Team Software Process,[SM] or TSP[SM]) in both academic and industrial settings is documented in numerous technical reports and peer-reviewed journal articles. Since PSP relies heavily on the collection and analysis of personal data as proof of effective process implementation, the claims made in these reports and articles are supported by objective, hard-data evidence.

[SM] Team Software Process and TSP are service marks of Carnegie Mellon University.

The concepts and methodologies of the PSP and TSP technologies have reached a level of maturity that warrant that further refinements be made by the professional community, academia, and certification entities. To support this effort, further educational expansion of the PSP must be accomplished by and accepted in the community. The research performed by Ford and Gibbs revealed that as a profession advances, it must have ways to assess and assure the adequacy of education and training curricula and the competency of individual professionals to further the profession [Ford 96]. Professional PSP competency measures are needed to assess both the level of knowledge acquisition and the level of skill in applying that knowledge. Certification is one of the most widely used mechanisms that a profession employs to make explicit the core set of knowledge and skills that a professional is expected to master, to establish objective assessments of those core competencies, and to provide a foundation for continuing qualification of individual professionals.

At the core of the process of maturing a profession is the establishment of a body of knowledge (BOK). A *body of knowledge* is a document generated by experts or masters in a particular profession to identify and delineate the concepts, facts, and skills that professionals and practitioners in that profession are expected to have mastered. The Institute of Electrical and Electronics Engineers (IEEE) Computer Society has established a body of knowledge for the software engineering profession as a whole. The PSP BOK document is meant to complement and build upon that work by describing the essential skills and knowledge specific to the PSP approach to software process improvement.

It is the authors' expectation that as PSP practice becomes more widespread, there will be further evolutions to the body of knowledge, particularly in the area of process extensions. The authors invite knowledgeable users of the PSP to submit suggestions and input for future revisions to this body of knowledge.

1.1 PURPOSE

The PSP BOK is not intended to be a comprehensive overview of the entire field of software engineering, nor is it meant to exhaustively delineate every supporting detail of the various key concepts and key skills that compose the PSP competency areas. Rather, the purpose of this document is to provide an overview of the competencies, knowledge areas, and key concepts and skills that constitute the core PSP body of knowledge. The main purposes of this document are to

- define the essential knowledge and skills that PSP-trained software engineering professionals are expected to master
- characterize the standard practices of PSP-trained software engineering professionals
- delineate the knowledge and skills that set PSP practitioners apart from common software and other engineering practices
- establish a baseline for developing, assessing, and accrediting PSP courses and curricula throughout academia

- facilitate the establishment of PSP certification programs that are based on an established and agreed-upon standard knowledge and skills set
- provide employers with a baseline for assessing the software skills and capabilities of their engineers and product development teams

Another purpose of this document is to define and delineate the baseline knowledge and skill set upon which the Carnegie Mellon® Software Engineering Institute (SEI) certification program for the SEI-Certified PSP Developer is based.

Although the principles and skills of the PSP can and should be applied to every phase of the product life cycle, the primary concentration is on improving the outputs of the development phase of software projects. Therefore, there are categories of software and other engineering knowledge that are not represented in the PSP BOK (architecture, requirements definition, hardware design, etc.), although it is assumed that a proficient software professional will have some degree of familiarity with such topics. It is also assumed that individuals who have mastered the PSP possess certain knowledge and skills, such as the ability to write software in one or more recognized programming languages, and proficiency in basic mathematics and applying statistical methods. Since these knowledge and skill areas are prerequisites for using the PSP, an exhaustive description of these areas is not included in this body of knowledge.

Similarly, although the PSP BOK is meant to guide the design, development, implementation, and assessment of courses and curricula based in part or in whole on the knowledge and skills delineated in it, the PSP BOK is not intended to be a guide for curriculum or course development. Such activities require pedagogical knowledge and expertise outside the domain of this body of knowledge; therefore, this document is meant to guide only the *content*—not the *methodology*—of PSP instruction and training.

1.2 SOURCES AND INFLUENCES

In preparing this document, the authors examined a number of reports delineating bodies of knowledge from other professional disciplines. Of these, three body of knowledge guidebooks provided guidance and inspiration in terms of structuring the document and depicting the architectural hierarchy used to describe the PSP BOK. These guides were

- *A Software Engineering Body of Knowledge Version 1.0*, by Thomas B. Hilburn, Iraj Hirmanpour, Soheil Khajenoori, Richard Turner, and Abir Qasem
- IEEE Computer Society's *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, 2004 Version
- Project Management Institute's *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, Third Edition

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

The structure of this document mirrors the general topics and document flow used by Hilburn et al.; the IEEE *SWEBOK*, 2004 Version and *PMBOK Guide* were influential in determining the depiction of the hierarchy of components used in the description of the PSP BOK.

The content specific to the various competency areas of the PSP BOK contained in Section 4 of this document is derived from the PSP books written by Watts S. Humphrey and numerous reports, listed in the bibliography of this document.

1.3 DOCUMENT ORGANIZATION

This document is divided into five major sections.

- Section 1 provides background information and an overview of the intended purposes of and audience for this body of knowledge and the sources and influences that affected its development.
- Section 2 outlines possible applications for the PSP BOK by software development professionals, the software development industry, and academic institutions.
- Section 3 summarizes the structure of the hierarchy used to describe the content of the body of knowledge and provides operational definitions of terms used in the document.
- Section 4 outlines the competency and knowledge areas that make up the body of knowledge and delineates the key concepts and skills that make up each knowledge area.
- Section 5 contains the summary and conclusions.

2 Suggested Uses of the PSP BOK

The PSP BOK can be used in professional, industrial, and academic settings. For example, it can be used as a basis for certifying practitioners who have attained proficiency in all of the key concepts and skills that the body of knowledge comprises. This section discusses potential uses of the PSP BOK in detail.

2.1 USE BY SOFTWARE DEVELOPMENT PROFESSIONALS

The definitions of essential concepts and skills that compose the PSP BOK can assist software engineering professionals in assessing their own skills and proficiencies and in identifying areas in which they may need further improvement.

2.2 USE BY THE SOFTWARE DEVELOPMENT INDUSTRY

The PSP BOK can be used by employers who want to establish an objective baseline for assessing the software development skills and capabilities of their engineers and product development teams. By understanding software engineering best practices, the industry can implement improvement efforts within its organizations, thereby achieving higher quality products and better management of costs and schedules.

2.3 USE BY ACADEMIC INSTITUTIONS

The PSP BOK can assist academic institutions in updating software engineering or computer science curricula to reflect current software development practices used in industry. As employers begin to require that newly hired developers possess the SEI-Certified PSP Developer credential, academic institutions can begin to prepare students for the certification examination. Some institutions may choose to offer a PSP course, while others may choose to integrate PSP into several of their courses. In both cases, institutions can use the guidance provided by this BOK to ensure that the topics included on the certification examination are adequately presented.

Academic institutions are expected to be innovative in developing ways to prepare students to master the PSP BOK. Whether via traditional classroom courses, distance education, or other media, the instruction that academic institutions offer will provide a benchmark for industrial or commercial training programs based on the PSP BOK. Academic instruction in the BOK competencies, knowledge areas, key concepts, and key skill areas also provides a baseline for assessing the quality of instruction offered through industrial or commercial training or other such venues.

3 PSP BOK Structure and Terminology

3.1 STRUCTURE

The body of knowledge delineated in this document is organized in an architectural hierarchy in which the concepts and skills of the PSP are described and decomposed into three levels of abstraction. For the purpose of this model, the term *concept* is used to describe the intellectual aspects of the PSP content; that is, the information, facts, terminology, and philosophical components of the technology. The term *skill* refers to the ability of an engineer to apply concepts to the performance of a task. Together, the key concepts and key skills form a *knowledge area*, and related knowledge areas constitute a *competency area*.

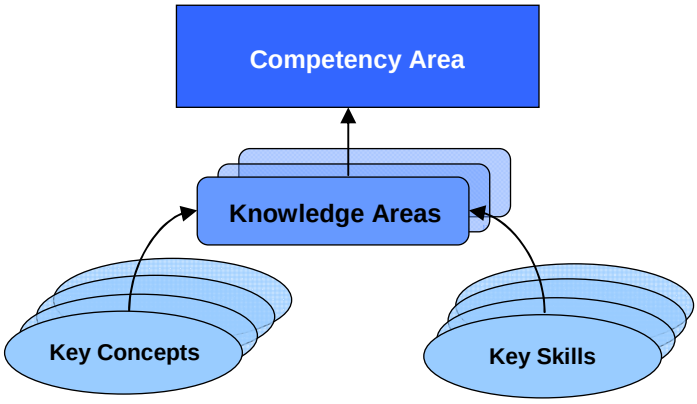

Figure 1: Architectural Hierarchy of the PSP BOK Components

3.2 OPERATIONAL DEFINITION OF TERMS

The PSP BOK uses the following terms to describe the categories of principles and processes it contains.

- competency area: a group of closely-related knowledge areas that a practitioner is well qualified to perform intellectually or physically
- knowledge area: the sum or range of specific understanding and ability gained through study of a set of concepts or through experience with a set of skills
- key concept: an explanatory principle applicable to a specific instance or occurrence within a particular knowledge area
- key skill: proficiency, facility, or dexterity that is acquired or developed through training or experience within a particular knowledge area

4 The PSP Body of Knowledge

This section contains a description of each major competency area, its supporting knowledge areas, and the key concepts and skills that compose each knowledge area. This information does not provide a detailed delineation of the PSP process but rather a high-level description of the proficiencies that a competent PSP-trained engineer is expected to master. As the PSP is adopted by broader audiences throughout the world, it is expected that the content of this BOK will evolve over time with an increased range of practice in a variety of environments and cultures.

The PSP BOK is composed of seven competency areas.

- Competency Area 1: Foundational Knowledge
- Competency Area 2: Basic PSP Concepts
- Competency Area 3: Size Measuring and Estimating
- Competency Area 4: Making and Tracking Project Plans
- Competency Area 5: Planning and Tracking Software Quality
- Competency Area 6: Software Design
- Competency Area 7: Process Extensions

The first two competency areas provide an overview of the foundation on which the PSP methods are built and an explanation of basic PSP concepts. Competency Areas 3, 4, 5, and 6 discuss more specific components such as planning, making and tracking schedules, measuring and improving product quality, and various techniques for designing software. The final competency area discusses advanced applications of the PSP by experienced practitioners.

Competency Area 1: Foundational Knowledge

Competency Area Number and Name	1. Foundational Knowledge
Competency Area Description	This competency area outlines the basic process definition and statistical methods (concepts and skills) on which the PSP is built.
Knowledge Areas	1.1 Process Definition 1.2 Process Elements 1.3 Statistics
References	[Humphrey 95, Chapters 1, 11, Appendix C] [Humphrey 05a, Chapters 2, 6, 13]

DESCRIPTIONS OF THE FOUNDATIONAL KNOWLEDGE KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
1.1 Process Definition	PSP is a series of defined processes that allow software engineers to produce high-quality products on time and within budget. This knowledge area outlines the concepts and skills needed to create, stabilize, and use defined processes.
1.2 Process Elements	This knowledge area describes the components that are included in any personal process and form a framework for organizing project work.
1.3 Statistics	Statistics are the foundation for PSP planning and tracking methodologies and also provide an objective means of analyzing and improving personal software engineering processes.

Knowledge Area 1.1: Process Definition

The PSP is a series of defined processes that allow software engineers to produce high-quality products on time and within budget. This knowledge area outlines the concepts and skills needed to create, stabilize, and use defined processes.

Key Concept Number and Name	Description
1.1.1 Prerequisite knowledge and skills (process description)	A *process* describes the sequence of steps that a knowledgeable engineer should follow to do a specified task. A process description should be brief and succinct without any of the tutorial or other explanatory material that would not be needed by a skilled and knowledgeable engineer.
1.1.2 Defined process	A *defined process* is a documented sequence of steps required to do a specific job. Processes are usually defined for jobs that are done repeatedly and need to be done in the same way each time that they are performed.

Key Concept Number and Name	Description
1.1.3 Rationale for defining a process	A defined process provides • a framework for planning, tracking, and managing work • a guide for doing the work correctly and completely, with the steps in the proper order • an objective basis for measuring the work and tracking progress against goals, and for refining the process in future iterations • a tool for planning and managing the quality of products produced • agreed-upon, mutually-understood procedures for team members to use in coordinating their work to produce a common product
1.1.4 Processes versus plans	A *process* is the defined set of steps for doing a task or project; a *plan* includes the process steps, plus other elements for a specific instantiation of that process, such as resources needed, roles of various project members, schedule, and budget.
1.1.5 Personal process	A *personal process* is a defined set of steps or activities that guide individuals in doing their personal work. It is usually based on personal experience and may be developed entirely "from scratch" or may be based on another established process and modified according to personal experience. A personal process provides individuals with a framework for improving their work and for consistently doing high-quality work.
1.1.6 Enactable process	An *enactable process* is a defined process that includes all of the elements required for using the process. An enactable process consists of a process definition, required process inputs, and assigned agents and resources (e.g., people, hardware, time, money).
1.1.7 Process phases	A defined process consists of a set of steps, elements, or activities that are generally called *phases*. Simple process phases consist of steps with no further substructure. More complex processes may have phases that are themselves processes. The steps or activities in each phase are defined by a *script* (see Key Concept 1.2.2). At a minimum, any process must have three phases: planning, development, and postmortem (see Key Concept 1.1.8).
1.1.8 PSP process phases	The basic PSP process has three phases. 1. Planning: Produce a plan to do the work. 2. Development: Perform the work. a. design the program b. review the design c. code the program d. review the code e. compile and fix all defects f. test the program and fix all defects 3. Postmortem: Compare actual performance against the plan, record process data, produce a summary report, and document all ideas for process improvement.

Key Concept Number and Name	Description
1.1.9 Incremental development with the PSP process	• The PSP facilitates incremental development. For larger projects, each increment can be an entire PSP project, a PSP development phase, or part of a PSP development phase, depending on the engineer's needs. • Various predefined PSP incremental development processes are available [Humphrey 05a]. • The PSP methods are used most effectively with large-scale incremental development when each increment is of high quality. If an earlier increment is of poor quality, regression testing and production of detailed test reports become high-priority activities in PSP (specifically, PSP3).

Key Skill Number and Name	Description
1.1.10 Define a high-quality process	Given a description of a product, define a high-quality personal process for building that product.

Knowledge Area 1.2: Process Elements

This knowledge area describes the components that are included in any personal process and form a framework for organizing project work.

Key Concept Number and Name	Description
1.2.1 Process elements	*Process elements* are components of a process. The PSP contains four basic elements: scripts, forms, measures, and standards.
1.2.2 Scripts	*Scripts* are expert-level descriptions that guide personal enactment of a process. They contain references to pertinent forms, standards, guidelines, and measures. Scripts may be defined at a high level for an entire process or at a more detailed level for a particular process phase. A process script documents • the process purpose or objective • entry criteria • any general guidelines, usage considerations, or constraints • phases or steps to be performed • process measures and quality criteria • exit conditions
1.2.3 Forms	*Forms* provide a convenient and consistent framework for gathering and retaining data. Forms specify the data required and where to record them. As appropriate, forms also define needed calculations and data definition. Paper forms may be used if automated tools for data gathering and recording are not readily available.

Key Concept Number and Name	Description
1.2.4 Measures	*Measures* quantify the process and the product. They provide insight into how the process is working by enabling users to • develop data profiles of previous projects that can be used for planning and process improvement • analyze a process to determine how to improve it • determine the effectiveness of process modifications • monitor the execution of their process and make next-step decisions • monitor ability to meet commitments and take corrective actions as needed
1.2.5 Standards	*Standards* provide precise and consistent definitions that guide the work and the gathering and use of data. Standards (such as coding, counting, and defect standards, and design review and code review checklists) enable measures to be applied uniformly across multiple projects and compared against each other.

Key Skill Number and Name	Description
1.2.6 Use PSP scripts	Be able to follow the script for a defined process and for all phases within that process. • Ensure that entry criteria have been satisfied. • Understand instructions for each step or phase and carry out the delineated activities. • Record data for time, size, defects, and schedule. • Ensure that the exit criteria have been satisfied.
1.2.7 Use PSP forms	Understand the purpose of each form and enter the necessary data correctly and accurately.
1.2.8 Use PSP measures	Derive and interpret measures correctly and accurately.
1.2.9 Use PSP standards	Define and interpret standards for • coding • size counting • defect type • design • reuse • any other area where standard behavior is known to be desirable

Knowledge Area 1.3: Statistics

Statistics are the foundation for PSP planning and tracking methodologies and also provide an objective means of analyzing and improving personal software engineering processes.

Key Concept Number and Name	Description
1.3.1 Distributions in the PSP	In the PSP, a *distribution* is a set of numerical values that are generated by some common process (actual sizes of parts developed or size estimates).

Key Concept Number and Name	Description
1.3.2 Mean in the PSP	The *mean* is the average value of a distribution. In the PSP, the mean is typically an estimate of the mean of the distribution, not the actual mean.
1.3.3 Variance in the PSP	*Variance* is a measure of the spread or tightness of a distribution around the mean. In the PSP, the variance is typically an estimate of the variance of the distribution, not the actual variance.
1.3.4 Standard deviation in the PSP	*Standard deviation* measures the amount of fluctuation in an estimate; it is the square root of the variance. Standard deviation is used in PSP in one method for categorizing software size into relative size tables. Standard deviation is also used as part of the calculation of prediction intervals.
1.3.5 Correlation in the PSP	*Correlation* is a measure of the degree to which two sets of data are related. In the PSP, correlation is measured between estimated and actual size and between estimated size and actual effort.
1.3.6 Significance of a correlation in the PSP	*Significance* measures the probability that two data sets have a high degree of correlation by chance. Estimates of size and effort in the PSP are more reliable when based on historical data that have a high degree of correlation that is significant.
1.3.7 Linear regression in the PSP	*Linear regression* determines the line through the data that minimizes the variance of the data about that line. When size and effort are linearly related, linear regression can be used to obtain projections from estimates.
1.3.8 Prediction interval in the PSP	The *prediction interval* provides the range around an estimate made with linear regression within which the actual value will fall with a certain probability. For example, in PSP, the 70% prediction interval for an estimate of size or time implies a 0.7 probability that the actual value of size or time will be within the range defined by the prediction interval.
1.3.9 Multiple regression in the PSP	*Multiple regression* is used in the PSP when estimations of size or time depend upon more than one variable. For example, if modifications to programs require much more time than additions, then “added” and “modified” can be separated into two variables for the regression calculation.
1.3.10 Standard normal distribution in the PSP	The *standard normal distribution* is a normal distribution translated to have a mean of zero and standard deviation of one. The standard normal distribution is used in the PSP when constructing a size estimating table.
1.3.11 Log-normal distribution in the PSP	Many statistical operations assume that data values are normally distributed, but some PSP measures do not meet this requirement. For example, size values cannot be negative but can have small values that are close to zero. When a log transformation is applied to data sets of this type, the resulting distribution may be normally distributed and, therefore, suitable for statistical analyses that assume normally distributed data. Statistical parameters for the normal distribution may be calculated and then transformed back to the original distribution. Size data in the PSP are generally log-normally distributed, so they must be transformed into a normal distribution for construction of a size estimating table.

Key Concept Number and Name	Description
1.3.12 Degrees of freedom in the PSP	*Degrees of freedom (df)* is a measure of the number of data points (n), as compared to the number of parameters (p) that are used to represent them. In linear regression, two parameters (β_0 and β_1) describe the line used to approximate the data. Since at least two points are needed to determine a line, the number of degrees of freedom is *n-2*. In general, the number of degrees of freedom is *n-p*.
1.3.13 t-distribution in the PSP	The *t-distribution* enables estimation of the variance of a normal distribution when the true value is not known, thus enabling calculation of statistical parameters based upon sample data. It is bell-shaped and varies depending upon the number of points in the sample. For fewer data points, the distribution is short with fat tails. As the number of data points increases, the distribution becomes taller with smaller tails and approaches the normal distribution. In PSP, the t-distribution is important because it helps to determine the significance of a correlation and the prediction interval for regression, each of which is dependent upon the number of points in the sample data set.

Competency Area 2: Basic PSP Concepts

Competency Area Number and Name	2. Basic PSP Concepts
Competency Area Description	This competency area describes the process improvement concepts and skills on which the PSP is built.
Knowledge Areas	2.1 Process Fidelity 2.2 Data Collection 2.3 Data Analysis 2.4 Process Improvement
References	[Feiler 92] [Humphrey 95, Chapters 1, 2, 7, 13] [Humphrey 05a, Chapters 1, 2, 13] [Humphrey 05b, Chapters 4-9] *TSP: Leading a Development Team*, Chapter 11[1]

DESCRIPTIONS OF THE BASIC PSP CONCEPTS KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
2.1 Process Fidelity	*Process fidelity* is the degree to which engineers follow their own defined personal process [Feiler 92]. This knowledge area addresses the effect of process fidelity on product quality.
2.2 Data Collection	This knowledge area describes the four basic PSP measures.
2.3 Data Analysis	This knowledge area describes the knowledge and skills needed by PSP engineers to analyze the process data that they collect.
2.4 Process Improvement	This knowledge area describes the knowledge and skills needed by PSP engineers to improve their own defined personal process.

Knowledge Area 2.1: Process Fidelity

Process fidelity is the degree to which engineers follow their own defined personal process. This knowledge area addresses the effect of process fidelity on product quality.

Key Concept Number and Name	Description
2.1.1 Process fidelity and useful data	In order to have meaningful data for implementing and improving a personal process, the process must be followed as defined.
2.1.2 Process fidelity and product quality	The quality of the product is governed by the quality of the process used to develop it. It is not enough to define a high-quality process; an engineer also must follow that process when developing the product. Having and consistently following a high-quality process will result in the production of high-quality products.

[1] Humphrey, Watts S. *TSP: Leading a Development Team.* Reading, MA: Addison-Wesley, to be published.

Key Concept Number and Name	Description
2.1.3 Process fidelity and planning	The project plan and committed delivery date are based on the process that will be used to develop the product. If the defined process is not followed, the plan no longer relates to what is being done, and plan progress can no longer be tracked accurately. Precise project tracking requires accurate data.
2.1.4 Process fidelity and performance improvement	A defined process with measures gives engineers the knowledge that they need to select the methods and practices that best suit their particular abilities and the tasks that they need to perform. Engineers must personally use well-defined and measured processes in order to consistently improve their performance.

Key Skill Number and Name	Description
2.1.5 Check entry criteria	Given a process script and a scenario description based on that script, determine if all entry criteria have been met.
2.1.6 Follow process steps	Given a process script and a scenario description based on that script, determine if the process was accurately executed as delineated by the script, and identify any process deviations.
2.1.7 Check exit criteria	Given a process script and a scenario description based on that script, determine if all exit criteria have been met.
2.1.8 Determine next steps/activities	Given a set of process scripts and a scenario describing a project, determine what steps/activities should be included in the next phase of a project, based on the phases or activities already completed and those that have yet to be done.
2.1.9 Phases vs. steps/activities	Given a process script and a scenario description based on that script, determine whether the action described is a process phase or a process step/activity.

Knowledge Area 2.2: Data Collection

This knowledge area describes the four basic PSP measures.

Key Concept Number and Name	Description
2.2.1 Basic PSP measures	The basic PSP measures are time, size, quality, and schedule data.
2.2.2 Time data	Time is measured in minutes and is tracked while doing the work. Basic components are *start date and time, end date and time, interrupt time,* and *delta time.*
2.2.3 Interrupt time	*Interrupt time* is not included in the time measurement for a task or process phase. If there is an interruption during the work, that time is subtracted from the time measurement.
2.2.4 Delta time	*Delta time* is the actual time that it took to complete a task or process phase. It is calculated as end time minus start time (less any interrupt time).
2.2.5 Time in phase	The *time in phase* is the planned or actual time spent in a particular PSP phase.
2.2.6 Size measures	A *size measure* is used to measure how big a work product is. Size measures are selected so that they are appropriate to the programming task and language (see Knowledge Areas 3.1 and 3.2).

Key Concept Number and Name	Description
2.2.7 Quality (defect data)	In PSP, product quality is measured in terms of defects. A *defect* is anything in the program that must be changed for it to be properly developed, enhanced, or used. Defects can be in the code, designs, requirements, specifications, or other documentation. Defects should be recorded as soon as they are discovered. To manage defects, engineers need to record the following data for *every* defect injected: date when the defect was discovered, phase when the defect was injected, phase when the defect was removed, defect type, time to find and fix the defect, and a brief description of the defect. A new defect may be injected while fixing another defect. In this case, the second defect is recorded separately, with a reference (called the fix reference) back to the original defect. Time for fixing each defect is recorded separately.
2.2.8 Defect type standard	The *defect type standard* defines categories into which similar defects can be placed. Consistent assignment of similar defects to the same defect type category is essential for process analysis.
2.2.9 Schedule measures	*Schedule measures* are used to plan when the project should be complete and to track actual progress against the plan.
2.2.10 Schedule data	See Competency Area 4.
2.2.11 Derived measures	The PSP provides a set of performance and quality measures to help engineers implement and improve their personal processes. Specific derived measures are discussed in later knowledge areas.

Key Skill Number and Name	Description
2.2.12 Track task or phase time	Accurately, completely, and consistently track time spent on a development project task or development phase. Missed entries should be recorded as soon as possible after the omission is discovered.
2.2.13 Track product size	See Knowledge Area 3.1.
2.2.14 Track defect data	Accurately, consistently, and completely track defects injected and removed while working on a development effort. Missed entries should be recorded as soon as possible after the omission is discovered.
2.2.15 Track schedule data	See Knowledge Area 4.5.
2.2.16 Track to-date data	Accurately, consistently, and completely track to-date time per phase, to-date reuse, to-dated added and modified, to-date total, to-date total new reusable size, to-date defects injected per phase, and to-date defects removed per phase.
2.2.17 Track to-date percent data	Accurately, consistently, and completely track to-date percent time per phase, to-date percent defects injected per phase, and to-date percent defects removed per phase.

Knowledge Area 2.3: Data Analysis

This knowledge area describes the knowledge and skills needed by PSP engineers to analyze the process data that they collect.

Key Concept Number and Name	Description

2.3.1 Measurement framework and data analysis	All of the PSP measures are related. Engineers must understand how each measure relates to the others and how they can be used to derive measures that provide information on process effectiveness.
2.3.2 Postmortem	A *postmortem analysis* of the work conducted at the completion of a phase or project provides valuable information, including • updated project data for time, size, defects, and schedule (actual, to-date, and to-date %) • updated calculations for quality or performance data • a review of actual performance against the plan • updated historical databases for size and productivity • process adjustments needed, based on personal data (notes made on process improvement proposal (PIP) forms, changes in design or code review checklists indicated by defects that escaped a phase, and so on).
2.3.3 Performance measures	The key performance measures of the personal process are • ability to meet schedule commitments for delivery of promised content • quality of delivered content • project-specific measures
2.3.4 Performance baselines	Before engineers can improve their performance, they first must understand the level of their current performance. After gathering enough project data to provide a meaningful amount of information for analysis, engineers conduct a baseline analysis of their to-date performance and formulate appropriate process changes to improve their performance in problem areas.

Key Skill Number and Name	**Description**
2.3.5 Combine PSP measures	Understand how measures can be combined to provide useful data for future project plans and process improvements. An example of this skill would be creating a chart based on estimated size vs. actual size across multiple projects to provide data for future size estimates.
2.3.6 Analyze historical data	Examine historical data to determine whether the correlation is adequate and significant as a basis for project and process measurement and analysis.
2.3.7 Understand data in relation to domains	Determine whether a given data set is appropriate for analysis. For example, data from projects based on the C++ language may not provide an appropriate correlation for analyzing projects based on the Ada language.

Key Skill Number and Name	Description
2.3.8 Postmortem analysis	Complete a postmortem analysis. • Update project data (actual, to-date, and to-date percent). • Calculate relevant quality or performance data. • Review actual performance against the plan. • Update historical databases with size and productivity data. • Document any PIPs generated during the phase or project. • Use personal data to make process adjustments. (For example, update design and code review checklists based on defects found in unit test.)
2.3.9 Determine the schedule performance baseline	Analyze the trend for on-time completions with the committed content.
2.3.10 Determine the quality performance baseline	Analyze the quality trend of delivered projects.
2.3.11 Analyze the schedule commitment performance	Determine if the on-time performance is acceptable. If it is not acceptable, identify possible root causes, such as inaccurate estimates for effort or available work time.
2.3.12 Analyze quality performance	Determine if the quality performance is acceptable. If it is not acceptable, perform a detailed quality analysis (as specified in Competency Area 5) and generate PIPs.
2.3.13 Analyze size-estimating accuracy	Analyze the historical personal process data for estimated size vs. actual size. Determine possible causes for misestimates. Create PIPs to address consistent problems. Consider the following questions. • How often is the estimate vs. actual within the 70% prediction interval? • Is there a tendency to miss parts in the conceptual design? • Is there a tendency to misjudge relative sizes of parts? • Are the size estimates improving over time?
2.3.14 Analyze effort-estimating accuracy	Analyze the historical personal process data for estimated effort vs. actual effort. Determine possible causes for misestimates. Create PIPs to address consistent problems. Consider the following questions. • How often is the estimate vs. actual within the 70% prediction interval? • Do the size estimate errors correlate with the effort estimate errors? • Do underestimated projects correlate with a larger percentage of rework? • Are the effort estimates improving?
2.3.15 Analyze size and time relationships	Analyze historical personal process data to determine any relationship between size and effort. Consider the following questions. • Is productivity stable? Why or why not? • Are there any quantitative differences between higher and lower productivity projects? If so, what might explain these quantitative differences?

Key Skill Number and Name	Description
2.3.16 Analyze phase yields	Analyze historical personal process data for phase yields. • Is there a relationship between yield and review rate (size reviewed per hour) for design and code reviews? • Are sufficient defects being found in the proper phases? • Which phases need the most improvement? Generate PIPs for possible improvements.
2.3.17 Analyze defects injected per phase	Generate a Pareto analysis of defect types and use it to analyze historical personal process data for defects injected per phase. • Determine which types of defects occur most often. • Determine which types of defects take longest to find and fix. • Analyze the per-phase and overall trends for defects injected per size unit. • Analyze the per-phase and overall trends for defects injected per hour. • Based on the analysis results, generate PIPs for possible defect prevention strategies.
2.3.18 Determine the cost of rework	Analyze data to determine the cost of rework. • Determine the percentage of PSP project time that defect-free testing would take. • Determine how long testing takes for PSP projects. • Determine what types of defects cost the most in terms of time to find and fix (per phase and per project). • Determine the types of defects most commonly found in personal compiling and testing. • Determine the types of defects most commonly found in product testing and in the delivered product. • Generate a Pareto analysis to identify the phases in which the defects found in the product were injected. • Generate PIPs to improve defect prevention and phase yields.
2.3.19 Determine high-leverage improvements	For each PIP generated, determine the estimated effort to implement that PIP and estimate its impact on performance. Then, select the highest leverage PIPs for implementation.

Knowledge Area 2.4: Process Improvement

This knowledge area describes the knowledge and skills needed by PSP engineers to improve their own defined personal process.

Key Concept Number and Name	Description
2.4.1 Goals for process improvement	The goals of process improvement are to improve the predictability and quality of delivery, while maintaining (or even improving) productivity.
2.4.2 Process changes	Large process improvement breakthroughs are rare, but small changes can be made almost every time a process is used.

Key Concept Number and Name	Description
2.4.3 Inspiration can strike at any time	Ideas for process improvement can occur at almost any time during project execution. Keep the PIP form at hand to record ideas before they are lost.
2.4.4 Set measurable performance goals	PSP practitioners should study their performance baselines and determine the most important areas for improvement. It is important to set measurable performance improvement goals (e.g., "reduce cost of rework by 20%") to know when the desired improvement has been achieved.
2.4.5 Use data to identify root causes of problems	PSP practitioners should analyze historical process data to determine root causes of past performance problems before implementing any process changes.
2.4.6 Implement highest payoff improvements first	Personal data analysis generates many PIPs. Practitioners should analyze their data and choose to implement the PIPs that offer the highest potential improvement in comparison to the effort required to make the changes.
2.4.7 Be aware of the result of process changes	PSP engineers use personal processes as the basis for doing their work. Practitioners must understand how to update their processes and be aware of the impact that changes may have on the applicability of their historical process data to future work based on the altered process.
2.4.8 After making process changes, monitor the performance results	To determine if implemented process improvements have been effective, PSP practitioners should periodically repeat the steps for baselining their work processes and comparing the baseline performance to previously established improvement goals. When so doing, practitioners should be careful to avoid the complications of bolstering and clutching. • *Bolstering* is the selective recall of only those results that reinforce an opinion or belief, usually manifest by forgetting failures and remembering only successes. Use of *all* PSP data from all projects should preclude bolstering. • *Clutching* is the tendency to perform badly when under pressure or when a good outcome is especially critical, thereby negating successful performance on past projects when using the same processes. By following established processes and using data (rather than instinct) as a basis for instantiating process changes, clutching can be minimized or avoided.
2.4.9 Watch for improvement opportunities	When working on PSP projects, practitioners should watch for new problem areas and be aware of ideas for continued improvement.
2.4.10 The PSP process improvement mechanism	The PSP uses a PIP form to capture problems with using the process and suggestions for improving or modifying it. Keep the PIP form at hand at all times for recording insights into opportunities for process improvement before those insights are lost.

Key Skill Number and Name	Description
2.4.11 Set realistic and challenging goals	Use data analysis to • identify problem areas • determine the root cause of problems • determine the potential effects of process change • set quantifiable and challenging improvement goals
2.4.12 Identify process changes	Develop PIPs that demonstrate insight into the strengths and weaknesses of the actual personal process.
2.4.13 Tailor the existing process	Update a defined process based on results of data analysis and generating PIPs.

Competency Area 3: Size Measuring and Estimating

Competency Area Number and Name	3. Size Measuring and Estimating
Competency Area Description	This competency area describes the size measurement and estimating concepts on which the PSP is built. The essential elements of size measurement and estimating are the ability to define appropriate size measures and to use disciplined methods and historical data to estimate size.
Knowledge Areas	3.1 Size Measures 3.2 Size Data 3.3 Size Estimating Principles 3.4 Proxies 3.5 The PROBE Estimating Method 3.6 Combining Estimates 3.7 Size Estimation Guidelines
References	[Humphrey 95, Chapters 4, 5, Appendix A] [Humphrey 97, Chapters 6, 10] [Humphrey 00] [Humphrey 05a, Chapters 3, 5, 6] *TSP: Leading a Development Team*[2]

DESCRIPTIONS OF THE SIZE MEASURING AND ESTIMATING KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
3.1 Size Measures	This knowledge area outlines the objectives of measuring size, the criteria for selecting a size measure, and the PSP size accounting system.
3.2 Size Data	This knowledge area discusses the primary ways that size data are used in the PSP.
3.3 Size Estimating Principles	This knowledge area discusses the principles upon which the PSP size estimating process is based. The PSP supports many size estimating methods, but all methods must adhere to these principles.
3.4 Proxies	This knowledge area discusses selecting and organizing proxy data.
3.5 The PROBE Estimating Method	The PSP uses a defined estimating process called *PRO*xy *B*ased *E*stimating (PROBE). This method is used to estimate both size and effort. This knowledge area defines how size estimates are made using the PROBE method.
3.6 Combining Estimates	This knowledge area discusses the various ways that estimates can be combined.
3.7 Size Estimation Guidelines	This knowledge area discusses the limitations of size estimating.

[2] Humphrey, Watts S. *TSP: Leading a Development Team.* Reading, MA: Addison-Wesley, to be published.

Knowledge Area 3.1: Size Measures

This knowledge area outlines the objectives of measuring size, the criteria for selecting a size measure, and the PSP size accounting system.

Key Concept Number and Name	Description
3.1.1 Rationale for using size measures	Objectives for using size measures include • achieving consistency in describing size • normalizing time and defect data • making better size estimates and plans
3.1.2 Types of measures	Measures may be categorized as • explicit or derived • objective or subjective • absolute or relative • dynamic or static • predictive or explanatory
3.1.3 Criteria for size measures	Useful size measures must be • related to development effort – Does the size of the product statistically correlate with development effort? – Does time spent on development of the measured part of the product represent a significant part of the project's work? • precise • machine countable • suitable for early planning
3.1.4 Counting standards	Counting standards provide guidance that is • precise about what to count • application/language specific • invariant, providing the same outcome each time the standard is applied
3.1.5 Physical and logical size	A *physical size measure* provides information about the size of a physical entity (the actual number of occurrences of an item in some product). A *logical size measure* also provides size information but relies on counting groupings of physical entities that can logically be grouped together. Physical size measures are based on a simple objectively described standard – a number that is arrived at no matter who is counting. The logical size measure of a physical entity does not necessarily correspond to the physical size measure of that same entity, depending on the counting standard defined for the logical measurement

Key Concept Number and Name	Description
3.1.6 Size accounting	PSP size accounting methods for planned, actual, and to-date size define the measures for • base (B): initially unmodified program to which subsequent enhancements are added • added (A): code that is added to the base code • modified (M): the part of the base code that is changed • deleted (D): the part of the base code that is subsequently removed • reused (R): an existing part or item that is copied unchanged from a source other than the base • added and modified (A&M): all added and modified code • new reusable (NR): a part or item that is developed with the intention of later reusing that part or item • total (T): the size of the entire program

Key Skill Number and Name	Description
3.1.7 Use the size measure selection procedure	Follow these steps for selecting a size measure. 1. Gather product development data (resources required, product characteristics measures, any special development conditions, etc.). 2. Rank the products by required resources. 3. Identify the characteristics that distinguish the products that took the greatest effort from those that required the least effort. 4. Select a size measure or size measures. For the candidate size measure(s) determine correlation. If there is no correlation, repeat steps 3 and 4 for other candidate size measures. Sample size measures include • test scenarios • requirements pages • database fields, tables, or records • lines of code
3.1.8 Produce and use a size counting standard	Follow the steps given in Key Skill 3.1.7 to determine the size measure(s) appropriate to the product being produced and satisfy 3.1.3. Then produce a size counting standard that satisfies 3.1.4.
3.1.9 Produce and use a coding standard	Create a coding standard that enforces the counting standard and includes good coding practices.
3.1.10 Calculate planned total size	Given plan data on base, added, modified, deleted, reused, and new reusable size, calculate plan total size as follows. *Plan total = A&M+B-M-D+R*
3.1.11 Calculate actual added and actual added and modified size	Given actual data on total, base, modified, deleted, and reused size, calculate actual added and actual added and modified size. • *Actual A = T-B+D-R* • *Actual A&M = A+M*
3.1.12 Calculate to-date size	Given historical size data for all projects to date, generate cumulative to-date reused, added and modified, total, and new reusable sizes.

Key Skill Number and Name	Description
3.1.13 Categorize measures by type	Given a PSP measure, categorize the measure according to its type. Example demonstrations of this skill include the following. • Recognize that “compile errors” is an objective measure. • Recognize that “defects” is an explicit measure. • Recognize that “defects/KLOC” is a derived measure.

Knowledge Area 3.2: Size Data

This knowledge area discusses the primary ways that size data are used in the PSP.

Key Concept Number and Name	Description
3.2.1 Size data help to make better plans	Size and time are often correlated, and when they are, size estimates can be used to estimate effort. Plans can then be created based on the size and effort estimates.
3.2.2 Size data are useful for tracking development effort	Size measures relate the effort expended to the amount of product produced, inspected, tested, etc.
3.2.3 Size data help in assessing program quality	Normalizing defect data based on size permits determination of the • quality of all or some part of the development process • relative defect content of some parts of large programs • future workload for maintenance and support

Knowledge Area 3.3: Size Estimating Principles

This knowledge area discusses the principles upon which the PSP size estimating process is based. The PSP supports many size estimating methods, but all methods must adhere to these principles.

Key Concept Number and Name	Description
3.3.1 Estimating is uncertain	No one knows how big the product will be, and the earlier in the process that the estimate is made, the less is known. Estimating can be biased by business needs and other pressures.
3.3.2 Estimating is a learning process	Estimating improves with experience and with data.
3.3.3 Estimating is a skill	Some people will be better at estimating than others. Most people improve at estimating with practice.
3.3.4 Strive for consistency	The objective of the size estimating process is to consistently follow a process that produces unbiased estimates. Doing so will balance the over-estimates and under-estimates.
3.3.5 Use defined methods for making estimates	Using a defined size estimating process facilitates learning, provides a framework for using historical data, and helps to remove bias from the process.
3.3.6 Estimates are subject to error	Estimating accuracy will fluctuate around some mean. Estimates may also have some bias.
3.3.7 Estimate in detail	When estimating in parts, the total error will be less than the sum of the part errors, assuming that the parts are estimated independently. Estimating in detail also helps to ensure that the estimate is complete.
3.3.8 Use historical data to make	When making size estimates, find a way to use whatever historical data

Key Concept Number and Name	Description
estimates	are available.

Knowledge Area 3.4: Proxies

This knowledge area discusses selecting and organizing proxy data.

Key Concept Number and Name	Description
3.4.1 Using proxies instead of a size measure	Most size measures that meet the required criteria are not available during planning. A *proxy* is a stand-in measure that relates product size to planned function and provides a means in the planning phase for judging (and therefore, of estimating) a product's likely size.
3.4.2 Criteria for choosing a proxy	A good proxy should • correlate with development costs • be easy to visualize at the beginning of a project • be a physical entity that can be measured and automatically counted
3.4.3 Using relative size tables	*Relative size tables* are used to organize proxy data so that historical proxy data can be used to estimate the size of similar new parts.
3.4.4 Building a relative size table	The PSP defines two procedures for building a relative size table from historical data: the *sort* method and the *standard deviation* method. Other methods may be used, but they must adhere to the size estimating principles.

Key Skill Number and Name	Description
3.4.5 Build a relative size table with the *sort* procedure	Separate the parts into functional categories such as calculation, text, data, etc. For each category 1. Sort the size data. 2. Pick the smallest value as very small (VS). 3. Pick the largest value as very large (VL). 4. Pick the median value as medium (M). 5. For large (L) and small (S), pick the midpoints between M and VL, and M and VS, respectively.
3.4.6 Build a relative size table with the *standard deviation* procedure	Separate the parts into functional categories such as calculation, text, data, etc. For each category 1. If the data are log-normally distributed, transform the data into a normal distribution by calculating the natural log of each datum; else skip this step. 2. Calculate the mean and standard deviation of the data set. 3. Convert the data to standard normal form. 4. Assign VS = -2; S = -1; M = 0; L = 1; VL = 2. 5. Apply the inverse of conversion to standard normal form to each of VS, S, M, L, and VL. 6. If the original data were log-normally distributed, calculate the anti-log of each of VS, S, M, L, and VL; else do nothing.
3.4.7 Assess the suitability of potential proxies	Given a product description and a list of potential proxies, use the criteria given in Key Concept 3.4.2 to assess the suitability of each proxy.

Knowledge Area 3.5: The PROBE Estimating Method

The PSP uses a defined estimating process called PROxy-Based Estimating (PROBE). This method is used to estimate both size and effort. This knowledge area defines how size estimates are made using the PROBE method.

Key Concept Number and Name	Description
3.5.1 What is PROBE?	*PROBE* is a procedure for estimating size and effort. The overall procedure is as follows. 1. Develop the conceptual design (see Key Concept 3.5.2). 2. Identify and size the proxies. 3. Estimate other elements. 4. Estimate program size. (Select the appropriate PROBE method.) 5. Calculate prediction intervals.

Key Concept Number and Name	Description
3.5.2 Conceptual design	The *conceptual design* is a high-level postulation of the product elements and their functions. The conceptual design subdivides a desired software component or system into its major parts. The conceptual design is a basis for size and effort estimation and may not necessarily reflect how the actual product is designed and implemented. It is produced solely as a basis for producing estimates (see Key Concept 4.2.4).
3.5.3 Formulate size estimates for proxies	Compare the size of new parts in the conceptual design against similar parts in the historical database to judge type and relative size. Use the number of items per part and historical size/part data to estimate proxy size.
3.5.4 Formulate estimates for various types of program elements	Count base size (B). Estimate modifications (M). Estimate deletions (D). Estimate base additions (BA). Estimate parts additions (PA). Estimate reused (R). Estimate planned new reusable (NR).
3.5.5 Select the appropriate PROBE method	• Check to see if method A can be used. – You have three or more data points (estimated E and actual A&M) that correlate. – The absolute value of β_0 is less than 25% of the expected size of the new program – β_1 is between 0.5 and 2. • If method A cannot be used, check to see if method B can be used. – You have three or more data points (plan A&M and actual A&M) that correlate. – The absolute value of β_0 is less than 25% of the expected size of the new program. – β_1 is between 0.5 and 2. • If method B cannot be used and you have historical data, use method C. • If you have no historical data, use method D.
3.5.6 Estimate program size	Calculate estimated proxy size, E = BA + PA + M. Calculate projected A&M size, P = β_0 + β_1(E), for methods A, B, and C. For method D, P = your professional judgment Calculate planned added size, A = A&M – M. Calculate planned total size, T = P + B – M + R.

Key Concept Number and Name	Description
3.5.7 Count and calculate actual data for various program elements	Count BA, PA, M, D, R. Calculate actual proxy size, E = BA+PA+M. Count actual total size, T. Calculate actual added size, A = T-B+D-R. Calculate actual added and modified size, A&M = A+M. Count actual new reusable, NR.
3.5.8 Prediction interval definition	The *prediction interval* is • the range within which the actual size is likely to fall 70% of the time • not a forecast • applicable only if the estimate behaves like historical data

Key Skill Number and Name	Description
3.5.9 Use the PROBE procedure	Produce a conceptual design. (This design is only a basis for planning; see Key Concepts 3.5.2 and 4.2.4.) 1. Identify and size the proxies. 2. Estimate other element sizes. 3. Use the appropriate PROBE method to estimate the program size and prediction interval.
3.5.10 Use PROBE method A	Check if PROBE method A can be used by assessing correlation, β_0, and β_1. If PROBE method A can be used, then calculate projected size as $y = \beta_0 + \beta_1(E)$, where • y = projected added and modified size • E = estimated proxy size • β_0 And β_1 are calculated using estimated proxy size and actual added and modified size
3.5.11 Use PROBE method B	If PROBE method A cannot be used, assess correlation, β_0, and β_1. If PROBE method B can be used, then calculate projected size as $y = \beta_0 + \beta_1(E)$, where • y = projected added and modified size • E = estimated proxy size • β_0 and β_1 are calculated using plan added and modified size, and actual added and modified size

Key Skill Number and Name	Description
3.5.12 Use PROBE method C	If PROBE methods A and B cannot be used, calculate project size as $y = \beta_0 + \beta_1(E)$, where • y = projected added and modified size • E = estimated proxy size • $\beta_0 = 0$ • $\beta_1 = \frac{ActualTotalAdded\ \&\ ModifiedSizeToDate}{PlanTotalAdded\ \&\ ModifiedSizeToDate}$
3.5.13 Use PROBE method D	If PROBE methods A, B, or C cannot be used, use your judgment to estimate added and modified size.
3.5.14 Calculate the upper prediction interval (UPI)	$UPI = PlanAdded\ \&\ Modified + Range(70\%)$
3.5.15 Calculate the lower prediction interval (LPI)	$LPI = PlanAdded\ \&\ Modified - Range(70\%)$

Knowledge Area 3.6: Combining Estimates

This knowledge area discusses the various ways that estimates can be combined.

Key Skill Number and Name	Description
3.6.1 Combine independent estimates	To combine independent estimates 1. Make separate linear regression projections. 2. Add projected sizes. 3. Add the squares of the individual ranges and calculate square root to calculate prediction interval.
3.6.2 Use multiple proxies	Use multiple regression when there is (*a*) correlation between development time and each proxy and (*b*) the proxies do not have separate size/hour data. 1. Identify and size each proxy. 2. Use multiple regression to project program size. $y = \beta_0 + x_1\beta_1 + x_2\beta_2 + ... + x_m\beta_m$ 3. Calculate prediction intervals. *UPI = projected size + range(70%)* *LPI = projected size – range(70%)*

Knowledge Area 3.7: Size Estimation Guidelines

This knowledge area describes the limitations of size estimating.

Key Concept Number and Name	Description
3.7.1 Clustered or grouped data	For data that are clustered or grouped, size estimates may not be very useful for estimating effort. However, the size estimate still may be useful in estimating average effort.

Key Concept Number and Name	Description
3.7.2 Extreme data points	Extreme data points can lead to erroneous β_0 and β_1 values, even with high correlation.
3.7.3 Unprecedented products	Resist making an estimate until the completion of a feasibility study and development of prototypes. Do not confuse making an estimate with guessing.
3.7.4 Data range	Estimates made for points outside the range of the data used to calculate β_0 and β_1 are likely to be seriously in error.

Key Skill Number and Name	Description
3.7.5 Determine if a set of historical data is useful for size estimation	Choose the appropriate PROBE method based on the type and quality of available data. Evaluate the data set for outliers, clustered or grouped data, and other factors that could affect the accuracy of the size estimate.

Competency Area 4: Making and Tracking Project Plans

Competency Area Number and Name	4. Making and Tracking Project Plans
Competency Area Description	This competency area discusses the ability to use an estimate of software size to plan and track a software project. Essential parts of project planning are the ability to construct a schedule, define tasks, plan tasks to conform to the schedule, and to track task completion against the plan.
Knowledge Areas	4.1 PSP Planning Principles 4.2 The PSP Planning Framework 4.3 Software Size and Effort 4.4 Task and Schedule Planning 4.5 Schedule Tracking with Earned Value 4.6 Planning and Tracking Issues
References	[Humphrey 05a, Chapters 4, 7]

DESCRIPTIONS OF THE MAKING AND TRACKING PROJECT PLANS KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
4.1 PSP Planning Principles	This knowledge area delineates the principles upon which the PSP planning framework is based.
4.2 The PSP Planning Framework	This knowledge area delineates the framework that integrates PSP planning tasks, historical databases, and tracking activities. It also addresses using PROBE to generate overall resource estimates.
4.3 Software Size and Effort	Project planning requires an estimate of software size (see Competency Area 3). This knowledge area describes the relationship between size and effort.
4.4 Task and Schedule Planning	This knowledge area describes how to use an overall resource estimate to create a schedule that defines the tasks to be completed and the expected completion dates.
4.5 Schedule Tracking with Earned Value	The PSP earned value (EV) system is used to track the progress of work completed against the schedule plan. This knowledge area discusses calculating EV, using the EV to determine work progress against the plan, and revising the planned schedule based on average EV earned to-date on the project.
4.6 Planning and Tracking Issues	Management must be kept informed of project status. Projects that will not be completed on schedule may need to be replanned.

Knowledge Area 4.1: PSP Planning Principles

This knowledge area delineates the principles upon which the PSP planning framework is based.

Key Concept Number and Name	Description
4.1.1 Plan your work	PSP principles state that the people who do the work are best suited to plan the work. • Engineers should always develop a plan for the work before committing to or starting a project. • When engineers are involved in developing the plan, they are most likely to be committed to that plan. Plans should be based on – a defined process – personal data
4.1.2 What is a PSP plan?	A PSP plan • defines the work and how it will be done • is a basis for agreeing on the cost and schedule for a project • is an organizing structure for doing the work • is a framework for obtaining the required resources • provides a record of what was initially committed
4.1.3 Detailed plans	Understand why detailed plans are needed.

Knowledge Area 4.2: The PSP Planning Framework

This knowledge area delineates the framework that integrates PSP planning tasks, historical databases, and tracking activities. It also addresses using PROBE to generate overall resource estimates.

Key Concept Number and Name	Description
4.2.1 Software product plan components	Components of a software product plan include the following. • Project sizing: How large is the project and how much time will be required to do the whole project? • Project structure: How will the work be accomplished? How should the tasks be sequenced? • Project status: What is the status of the project at any given time? How can the completion date be estimated? • Assessment: Compare the actuals to estimates. How good was the plan? How can the plan be improved next time?
4.2.2 PSP planning framework	The PSP planning framework consists of seven basic tasks. 1. Define the requirements (see Key Concept 4.2.3). 2. Produce the conceptual design (see Key Concept 4.2.4). 3. Produce the product size estimate (see Key Skill 3.5.9). 4. Produce the resource estimate (see Key Concept 4.2.6). 5. Produce the schedule (see Key Concept 4.2.10 and Knowledge Area 4.5). 6. Develop the product (see Key Concept 4.2.11). 7. Analyze the process (see Key Concept 4.2.12 and Key Concept 2.3.2).

Key Concept Number and Name	Description
4.2.3 Requirements definition	Start by defining the work that needs to done in as much detail as possible. The accuracy of the plan is dependent on how much the engineers know about the work to be done at the time when the work is being planned.
4.2.4 Produce the conceptual design	The conceptual design (see Key Concept 3.5.2) is a preliminary approach to the product that names the expected objects and their functions. When making a conceptual design, several alternative approaches might be considered in order to choose the optimal approach to doing the development work. For larger products, several steps may be needed to produce a conceptual design, starting with a system- or high-level product design, and then subdividing the resulting parts to a level of detail that corresponds to existing elements in the historical database (if any). Then use the PROBE methods to produce size and resource estimates.
4.2.5 Use PROBE for size and resource estimation	The PROBE method is used to estimate the size of the product and the time required to do the work (see Key Skill 3.5.9 and Key Concept 4.2.6).
4.2.6 Select the appropriate PROBE method for resource estimation	• Check to see if method A can be used. – You have three or more data points (estimated E and actual development time) that correlate. – The absolute value of β_0 is near 0. – β_1 is within 50% of 1/(historical productivity). • If method A cannot be used, check to see if method B can be used. – You have three or more data points (plan A&M and actual development time) that correlate. – The absolute value of β_0 is near 0. – β_1 is within 50% of 1/(historical productivity). • If method B cannot be used and you have historical data, use method C. • If you have no historical data, use method D.
4.2.7 To-date time in phase	*To-date time in phase* is the sum of the actual times for this project plus the to-date times in phase from historical projects.
4.2.8 To-date percent time in phase	*To-date percent time in phase* is the percentage of to-date time in each phase.
4.2.9 Distributing time across phases	Time is distributed across phases using historical to-date percent for time in phase.
4.2.10 Schedule projection	An earned value schedule provides a projection of the project completion date (see Knowledge Area 4.5).
4.2.11 Product development	Product development is guided by the defined personal process used to generate the plan. As the work is done, engineers gather and record data.
4.2.12 Process analysis	At the end of a project, the gathered data is analyzed (see Key Concept 2.3.2).

Key Concept Number and Name	Description
4.2.13 Cost performance index (CPI)	Calculate the cost performance index (CPI) as $\frac{\textit{planned total development time to date}}{\textit{actual total development time to date}}$

Key Skill Number and Name	Description
4.2.14 Use PROBE method A	Check if PROBE method A can be used by assessing correlation, β_0, and β_1. If PROBE method A can be used, then calculate projected development time as $y = \beta_0 + \beta_1(E)$, where • y = projected development time • E = estimated proxy size • β_0 And β_1 are calculated using estimated proxy size and actual development time
4.2.15 Use PROBE method B	If PROBE method A cannot be used, assess correlation, β_0, and β_1. If PROBE method B can be used, then calculate projected development time as $y = \beta_0 + \beta_1(E)$, where • y = projected development time • E = estimated proxy size • β_0 and β_1 are calculated using plan added and modified size and actual development time
4.2.16 Use PROBE method C	If PROBE methods A and B cannot be used, calculate project development time as $y = \beta_0 + \beta_1(E)$, where • y = projected development time • E = estimated proxy size • $\beta_0 = 0$ • $\beta_1 = \frac{ActualTotalDevelopmentTimeToDate}{PlanTotalAdded\ \&\ ModifiedSizeToDate}$
4.2.17 Use PROBE method D	If PROBE methods A, B, or C cannot be used, then use your judgment to estimate development time from the estimated added and modified size.
4.2.18 Calculate the UPI and LPI	See Key Skills 3.5.14 and 3.5.15.
4.2.19 Calculate to-date time in phase	Given historical time in phase data for all projects to date, calculate the cumulative to-date time in phase for each development phase.
4.2.20 Calculate to-date percent time in phase	Given historical time in phase data for all projects to date, calculate to-date percent time in phase for each development phase.
4.2.21 Distribute planned time across PSP project phases	Use to-date percent calculations for planned time in phase to distribute the total development time across the PSP project phases.
4.2.22 Use the cost performance index	Calculate the CPI and interpret the implications of the CPI with respect to future project performance.

Knowledge Area 4.3: Software Size and Effort

Project planning requires an estimate of software size (see Competency Area 3). This knowledge area describes the software relationship between size and effort.

Key Concept Name and Number	Description
4.3.1 Size and effort correlation	Larger programming projects require more effort. Accurately estimating programming effort requires use of a size measure that has a significant correlation with effort. Size data are suitable for planning purposes if the r^2 value is greater than 0.5 and if the tail area in the significance calculation is ≤ 0.05.
4.3.2 Productivity	*Productivity* is the ratio of a product's size to the time expended to develop that product, generally measured as size measure per hour.

Key Skill Number and Name	Description
4.3.3 Determine whether size and effort data are correlated	Analyze the correlation coefficient and the significance of the correlation to determine whether the size and effort data are correlated and can be used for planning purposes.

Knowledge Area 4.4: Task and Schedule Planning

This knowledge area describes how to use an overall resource estimate to create a schedule that defines the tasks to be completed and the expected completion dates.

Key Concept Number and Name	Description
4.4.1 Project plan characteristics	A project plan must be • accessible: easy to locate and reference • clear: straightforward and easy to read • specific: responsibilities and costs identified • precise: appropriate level of precision • accurate: based on relevant data and an unbiased estimating process
4.4.2 Period plans and project plans	A period plan covers a specific unit of time, such as a week or month. A project plan describes *all* efforts and costs for developing a product.
4.4.3 Task and working hours	*Task hours* is a measure of the time spent working on defined project tasks. *Working hours* includes task hours and accounts for non-task activities such as time reading and answering e-mail, attending meetings, etc.
4.4.4 Milestones	*Milestones* are key indicators of project progress. Their completion dates can be estimated so that progress against them can be tracked.
4.4.5 Schedule plan requirements	Required elements for producing a schedule plan are • a calendar of available time • the order in which the tasks are to be completed • estimated effort for each task
4.4.6 Task order	Task order is driven by the development strategy. • Each task needs completion criteria. • Task interdependencies must be defined.

Key Concept Number and Name	Description
4.4.7 Estimated task time	The time needed for task completion is estimated in one of several ways: • from the size of the product produced by that task and historic product data from similar tasks • from an overall estimate based on the to-date percent data from similar completed processes • using the appropriate PROBE estimating technique

Key Skill Number and Name	Description
4.4.8 Estimate task hours	Develop an estimate of task hours for a project. 1. List project tasks on the Task Planning template in the order in which the tasks should be completed. 2. Enter the number of hours estimated for each task. Use PSP data for similar completed tasks, when available.
4.4.9 Produce a PSP schedule plan	Produce a schedule plan for a PSP project. 1. Pick an appropriate time period (e.g., three to six months from the planned start date). 2. Distribute the estimated available task time over the duration of the project schedule. 3. Calculate cumulative planned schedule hours to the end of the project period.
4.4.10 Produce a PSP task plan	Produce a task plan for a PSP project. 1. Estimate the task hours (see Key Skill 4.4.8). 2. Calculate the sum of total planned task hours. 3. Calculate the plan time period in which each listed task will be completed, based on the schedule plan. 4. Calculate the planned completion date of the project.

Knowledge Area 4.5: Schedule Tracking with Earned Value

The PSP earned value system is used to track the progress of work completed against the schedule plan. This knowledge area discusses calculating EV, using the EV to determine work progress against the plan, and revising the planned schedule based on average EV earned to-date on the project.

Key Concept Number and Name	Description
4.5.1 Planned value (PV)	The *planned value* of a task is equal to its planned time expressed as a percentage of the total planned time for the project. For example, a 5-hour task in a 50-hour project would have a PV of 10.
4.5.2 Earned value (EV)	*Earned value* is a method used for tracking the actual progress of completed work against the overall project plan. As each task is completed, its PV is added to the cumulative EV for the project. Partially-completed tasks do not contribute to the EV total.
4.5.3 Using EV measures	When using EV, keep these limitations in mind. • The EV method assumes that the rate of task completion in the fu-

Key Concept Number and Name	Description
	ture will be roughly the same as it was in the past. If this is not the case, the EV projections will not be accurate. • The EV method measures progress relative to the plan. If the plan is inaccurate, the EV projections are also likely to be inaccurate. • The EV method assumes that the project's resources are uniform. If the staffing level increases, the EV projections will be pessimistic, and if the staffing is cut, the projections will be optimistic.
4.5.4 EV as a measure of actual progress relative to planned progress	At any time during a project, the sum of value earned for completed tasks represents the percentage of work that has been completed. A comparison of the cumulative EV to the cumulative planned EV at a given time indicates progress of the work against the planned schedule. • planned EV is the same as actual EV: work is on schedule • actual EV is larger than planned EV: work is ahead of schedule • planned EV is larger than actual EV: work is behind schedule
4.5.5 Project tracking with EV	During planning, the total PV for project tasks can be computed for each time period. Likewise, adding up the EVs for completed tasks at any time period in a project determines the percentage of completed work to-date for the project. At any point in the project, the actual EV earned can be compared to the cumulative planned EV to determine if the project is on schedule, behind schedule, or ahead of schedule.

Key Skill Number and Name	Description
4.5.6 Calculate PV for each task	Calculate PV for a task by dividing the estimated time ("planned time") for that task by the total planned time for all tasks, then multiplying the quotient by 100.
4.5.7 Calculate PV for each time period	Calculate the PV for a time period by adding the PVs for all tasks that are planned to complete during that time period.
4.5.8 Compute cumulative PV for a given time period	Calculate the cumulative PV to-date for a given time period by adding the PVs for all preceding time periods to the PV for the given time period.
4.5.9 Compare EV to-date against PV to-date	Compute EV for a given time period and the cumulative EV for that time period by using the same procedure for calculating PV. The cumulative EV can be compared to cumulative PV to determine if the project is on schedule.
4.5.10 Estimate the project completion date	Compute the estimated project completion date by calculating the average EV per week to-date and then using the average value for EV per week to compute the time necessary to complete the remaining planned value. This assumes that the project continues to earn the average EV rate as before.
4.5.11 Determine if the project is catching up or falling further behind	Compare the actual EV with the planned EV for the latest time period to show if the project is falling further behind or catching up.
4.5.12 Make an alternative estimate of project completion	Estimate the project completion date based on average EV to date, the EV in the latest time period, or some other value. Use of these various methods provides different insights into project status and progress.

Knowledge Area 4.6: Planning and Tracking Issues

Management must be kept informed of project status. Projects that will not be completed on schedule may need to be replanned.

Key Concept Number and Name	Description
4.6.1 Informing management of issues	Keep management informed of results from earned value analyses and inform them about schedule problems. Data about project status may be helpful in obtaining management assistance.
4.6.2 When to adjust a plan	A plan should reflect the way that the engineer is actually working. If it does not, the plan should be revised. When work methods or processes are revised, the entire plan should be re-examined.
4.6.3 Handling part-time assignments	Part-time assignments may be troublesome because task hours are divided among several projects. Frequent switching between tasks makes work on any one task difficult, and hampers coordination with other team members on a project.

Competency Area 5: Planning and Tracking Software Quality

Competency Area Number and Name	5. Planning and Tracking Software Quality
Competency Area Description	This competency area describes the need to produce products that satisfy users' needs, ways to measure the degree to which user needs are met, and ways to produce high-quality products.
Knowledge Areas	5.1 PSP Quality Principles 5.2 Quality Measures 5.3 Quality Methods 5.4 PSP Code Reviews 5.5 PSP Design Reviews 5.6 Review Issues
References	[Humphrey 05a, Chapters 8, 9]

DESCRIPTIONS OF THE PLANNING AND TRACKING SOFTWARE QUALITY KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
5.1 PSP Quality Principles	This knowledge area outlines the principles upon which the PSP quality framework is based.
5.2 Quality Measures	PSP data enable determination of measures of product and process quality and the effectiveness of the process at removing defects.
5.3 Quality Methods	Personal reviews are an effective and efficient way to improve program quality and programmer productivity. Various review methods are effective in different situations.
5.4 PSP Code Reviews	Code reviews should follow a defined process and employ checklists constructed from personal defect data. Consistency in following a review strategy based on experience can make reviews more efficient and effective.
5.5 PSP Design Reviews	Design reviews should follow a defined review process, including appropriate design analyses, using checklists that are built on sound design principles. Consistency in following a review strategy based on measured experience can make reviews more efficient and effective.
5.6 Review Issues	Reviews can be very effective if they are conducted using guidelines that are based on extensive and quantified experience.

Knowledge Area 5.1: PSP Quality Principles

This knowledge area outlines the principles upon which the PSP quality framework is based.

Key Concept Number and Name	Description
5.1.1 Personal responsibility	To produce quality products, engineers must feel personally responsible for the quality of their products (see Knowledge Area 7.4).

5.1.2 The economics of quality	• It costs less to find and fix defects earlier in a process, rather than later. • The longer a defect remains in a product, the greater the cost to remove it. • Testing is an inefficient and ineffective way to remove defects. • It is more efficient to prevent defects than to find and fix them. • The right way is always the fastest and cheapest way to do a high-quality job. • Code reviews are fundamentally more efficient than testing for finding and fixing defects.
5.1.3 Product quality	A software product must satisfy a minimum threshold of functionality and usefulness. The product should also satisfy user expectations with respect to a number of other criteria. • The product must work, i.e., perform with reasonable consistency. If this goal is not achieved, then nothing else is important. Additional user concerns might include – performance – safety – security – usability – compatibility – functionality • The product must provide functionality that the user needs and at the time the user needs it. In many development projects, users' perceptions of quality are frequently overlooked because developers spend most of their time finding and removing defects.
5.1.4 Process quality	A quality process must meet the needs of its users to produce quality products efficiently. A quality process must • produce a quality product consistently • be usable and efficient • be easy to learn and adapt to new circumstances

Knowledge Area 5.2: Quality Measures

PSP data enable determination of measures of product and process quality and the effectiveness of the process at removing defects.

Key Concept Number and Name	Description
5.2.1 Personal defect data	Personal defect data are useful in understanding and refining the personal process. Analysis of these data provides a valuable resource for constructing personal review checklists.
5.2.2 To-date defects injected and removed	*To-date defects injected and removed* is the sum of the actual defects injected and removed for each project phase, plus the to-date defects injected and removed per phase from historical projects.
5.2.3 To-date percent defects injected and removed	*To-date percent defects injected and removed* is the percentage of to-date defects injected and removed in each phase.
5.2.4 Yield	*Yield* is the percentage of defects in the program that are removed in a particular phase or group of phases.
5.2.5 Phase yield	*Phase yield* is the percentage of defects removed during a phase.
5.2.6 Process yield	*Process yield* is the percentage of defects removed prior to entering the compile phase (or before entering unit test if there is no compile phase).
5.2.7 Review yield	*Review yield* is the percentage of defects in the program found during the review.
5.2.8 Percent appraisal cost of quality (COQ)	*Percent appraisal COQ* is the percentage of development time spent in design and code review.
5.2.9 Percent failure COQ	*Percent failure COQ* is the percentage of development time spent in compile and test.
5.2.10 Cost of quality (COQ)	*Cost of quality* is the percentage of time spent performing appraisal and failure tasks.
5.2.11 COQ appraisal to failure ratio (COQ A/FR)	*COQ A/FR* is the ratio of time spent in appraisal tasks to time spent in failure tasks.
5.2.12 Defect density	*Defect density* is the number of defects found per size measure.
5.2.13 Process quality index (PQI) components	The *process quality index (PQI)* is one way to assess process quality. PQI has the following components. • phase-time ratios that describe the relationship between the time spent in one phase and time spent in another – The ratio of design time to coding time provides an indication of design quality. – The ratio of design review time to design time provides an indication of design review quality. – The ratio of code review time to coding time provides an indication of code review quality. • defect density measures – The ratio of compile defects to a size measure indicates code quality. – The ratio of unit test defects to a size measure indicates program quality.

Key Concept Number and Name	Description
5.2.14 Using the PQI	If one assumes a desired defect density of *x* in program use and that each defect removal phase has a yield of 50%, then • *2x* defects/size measure should be removed by system test • *4x* defects/size measure should be removed by integration test • *8x* defects/size measure should be removed by unit test • *16x* defects/size measure should be removed by compile For example, if the goal is less than 0.5 defects/size measure after system test, then the defect removal rate for the unit test phase should be 4 defects/size measure. (**Note:** The defect density measures may be modified if desired. In particular, if there is no compile phase, then defect density for another defect removal phase such as personal reviews can be used instead of those for the compile phase.)
5.2.15 The PQI	The PQI components are normalized to [0,1] such that zero represents poor practice and one represents desired practice. The ratios are plotted on the axes of a pentagon with scale [0,1]. The resulting polygon can be compared with the containing pentagon to determine the quality of the process. Recommended values are as follows. • Design quality: Design time should be at least 100% of coding time (1.0). • Design review quality: Design review time should be at least 50% of design time (0.5). • Code review quality: Code review time should be at least 50% of coding time (0.5). • Code quality: The number of compile defects/KLOC should be less than 10. • Program quality: For program code, the number of unit test defects/KLOC should be less than 5.
5.2.16 Composite PQI	A *composite PQI* measure represents the overall process quality; the measure is a product of the normalized PQI component values. See Appendix A for calculation.
5.2.17 Phase defect removal rate	*Phase defect removal rate* is the number of defects found per hour of phase time.
5.2.18 Review rate	*Review rate* refers to the size of product reviewed per hour.
5.2.19 Defect-removal leverage (DRL)	*Defect-removal leverage* is a measure of the relative effectiveness of defect removal for any two process phases. For example, the DRL for design review relative to unit test would be defined as follows. DRL(DR/UT) = defects per hour in design review divided by defects per hour in unit test

Key Skill Number and Name	Description
5.2.20 Calculate and interpret yields	Distinguish between phase yield and process yield. Calculate phase yield and process yield. Explain the quality implications of various yield values.
5.2.21 Calculate and interpret various COQ values	Calculate % Appraisal COQ, % Failure COQ, COQ, and COQ A/FR. Explain the quality implications of various COQ values for each value and for COQ values compared to other quality measures (i.e., yield).

Key Skill Number and Name	Description
5.2.22 Interpret the PQI	Interpret the quality implications of various PQI figures and values. For components with low PQI, propose strategies to address the quality risk.
5.2.23 Calculate and interpret review yield	Given program review data, calculate the review yield and describe its quality implications.
5.2.24 Calculate and interpret defect density	Given program defect data, calculate defect density and interpret its quality implications.
5.2.25 Calculate and interpret the defect removal rate	Given defect and time in phase data, calculate the defect removal rate and interpret its quality implications.
5.2.26 Calculate and interpret review rates	Given size and time in phase data, calculate the review rates and interpret their quality implications.
5.2.27 Calculate and interpret the DRL	Given defect removal rates, calculate the DRL values and interpret their quality implications.
5.2.28 Calculate to-date defects injected and removed	Given historical defects injected and removed data for all projects to date, calculate cumulative to-date defects injected and removed for each development phase.
5.2.29 Calculate to-date percent defects injected and removed	Given historical defects injected and removed data for all projects to date, calculate to-date percent defects injected and removed for each development phase.

Knowledge Area 5.3: Quality Methods

Personal reviews are an effective and efficient way to improve program quality and programmer productivity. Various review methods are effective in different situations.

Key Concept Number and Name	Description
5.3.1 Inspections	An *inspection* is a structured team review of a software product.
5.3.2 Walkthroughs	A *walkthrough* is less formal than an inspection. Design or code is presented to an audience that raises issues and asks questions.
5.3.3 Personal reviews	A *personal review* is conducted by the engineer who examines his or her own product with the goal of finding and fixing as many defects as possible. Personal reviews should precede any other activity that uses the product (coding, compiling, testing, inspecting, etc.).
5.3.4 Personal review principles	• Find and fix all defects in your work. • Use a checklist derived from personal defect data. • Follow a structured review process. • Follow sound review practices. • Measure your reviews. • Use data to improve your reviews. • Produce reviewable products. • Use data to prevent defects.
5.3.5 Relationship between reviews and inspections	A personal review of design or code should precede any inspection. A review before inspection assures inspectors that they are looking for more subtle design and coding issues, rather than obvious mistakes.

Key Skill Number and Name	Description
5.3.6 Use effective personal review practices	For effective and efficient reviews, these practices should be followed. 1. Take a break between working and reviewing. 2. Review products in hard copy form, rather than on a screen. 3. Check off each item as it is completed. 4. Update design and code review checklists periodically to respond to changes in personal data. 5. Build and use a different checklist for each design method or programming language. 6. Thoroughly analyze and verify every non-trivial design construct (see Knowledge Area 6.6).

Knowledge Area 5.4: PSP Code Reviews

Code reviews should follow a defined process and employ checklists constructed from personal defect data. Consistency in following a review strategy based on experience can make reviews more efficient and effective.

Key Concept Number and Name	Description
5.4.1 Code review checklist	A code review checklist is specific to an individual's coding process. It lists defect categories that have caused problems in the past so that these defects are checked for during the review.
5.4.2 PSP code review process	1. Obtain the code review checklist, coding standard, and defect standard. 2. Print a copy of the code to be reviewed. 3. For each defect category on the checklist, make a complete pass over the code and check off each item as it is completed. 4. Correct all defects and check each defect fix for correctness.
5.4.3 Code review strategy	A review strategy should be based on data that show the strategy to be effective. An initial strategy is to examine lower-level modules first so that procedural abstractions are reviewed and understood before higher-level ones. After a strategy has been determined to be effective, it should be followed consistently. Depending on the type of product and the engineer's knowledge of its design, different review strategies may be appropriate.
5.4.4 Review against a coding standard	Code reviews should check for compliance with coding standards. Applicable coding standards should be references in the code review checklist.

Key Skill Number and Name	Description
5.4.5 Build a code review checklist	Given a set of PSP defect data and a template, build a code review checklist.
5.4.6 Use a code review checklist	Given a code review checklist, coding standard, defect standard, and a copy of some code in a familiar programming language, use the code review checklist to perform a code review.

Knowledge Area 5.5: PSP Design Reviews

Design reviews should follow a design review process, including appropriate design analyses, based on checklists that are built on sound design principles. Consistency in following a review strategy based on measured experience can make reviews more efficient and effective.

Key Concept Number and Name	Description
5.5.1 Design review principles	• Produce designs that can be reviewed. • Follow an explicit review strategy. • Review the design in stages. • Verify that logic correctly implements the requirements. • Check for safety and security issues.
5.5.2 Design review checklist	A design review checklist is specific to an individual's design process. It lists defect categories that have caused problems in the past so that these defects are checked for during the review.
5.5.3 PSP design review process	1. Obtain the design review checklist and design and defect standards. 2. Print a copy of the design to be reviewed, if appropriate. 3. For each defect category on the checklist, make a complete pass over the design and check off each item as it is completed. 4. Correct all defects and check each defect fix for correctness.
5.5.4 Design review strategy	A review strategy should be based on data that show the strategy to be effective. After a strategy has been determined to be effective, it should be followed consistently.

Key Skill Number and Name	Description
5.5.5 Build a design review checklist	Given personal defect data, build a design review checklist.
5.5.6 Perform a design review	• Examine the program and checklist and decide on review strategy. • Use the design review checklist to perform a design review.
5.5.7 Verify the design	See Knowledge Area 6.6.

Knowledge Area 5.6: Review Issues

Reviews can be very effective if they are conducted using guidelines that are based on extensive and quantified experience.

Key Concept Number and Name	Description
5.6.1 Review efficiency	Design and code reviews find defects directly, helping the reviewer to gain a mental picture of the intended behavior of the program. In large system-development processes, design and code reviews are especially important because effective scale-up of PSP methods requires that all increments be of high quality. To ensure that large-scale systems achieve the same high quality as smaller systems, the PSP3 scripts should be followed, and each module and/or increment should be subjected to design reviews, code reviews, and regression testing to ensure that new increments do not cause problems with previously-tested and accepted functioning modules.
5.6.2 Reviewing before or after compiling	Many development environments use automatic code analyzers and/or compilers that are quite helpful; their use is not discouraged. However, to be the most effective, the review should be performed before using the code analyzer or compiler. Code reviews should be performed prior to testing.
5.6.3 Review objectives	Properly conducted code reviews significantly reduce testing time and produce high-quality results. Unless the engineer is committed to producing high-quality products, the review process is likely to be ineffective. Engineers whose objective is to begin testing as soon as possible rarely perform code reviews or perform them so poorly that they are a waste of time.

Competency Area 6: Software Design

Competency Area Number and Name	6. Software Design
Competency Area Description	This competency area describes the ability to incorporate design and design verification activities into a personal software development process.
Knowledge Areas	6.1 Software Design Principles 6.2 Design Strategies 6.3 Design Quality 6.4 Design Documentation 6.5 Design Templates 6.6 Design Verification
References	[Humphrey 95, Chapters 10, 12] [Humphrey 05a, Chapters 10-12]

DESCRIPTIONS OF THE SOFTWARE DESIGN KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
6.1 Software Design Principles	This knowledge area describes software design principles as incorporated in the PSP.
6.2 Design Strategies	Design is a complex intellectual process that cannot be reduced to a mechanical procedure, but some guidelines and strategies can be helpful. This knowledge area addresses the design strategies used in the PSP.
6.3 Design Quality	This knowledge area describes the key characteristics that can be used to assess the quality of a software design.
6.4 Design Documentation	Software designs must be documented, along with the related requirements, constraints, and rationale. This knowledge area discusses the design documentation that is the responsibility of the individual software engineer.
6.5 Design Templates	The PSP does not specify design techniques, but does provide a set of templates as a frame for design documentation. The templates help to ensure that a system and its modules are completely and precisely implemented.
6.6 Design Verification	To be effective, design reviews must go beyond simply reading through a design document. The PSP provides a number of design verification techniques that can be used to identify errors and omissions in software designs.

Knowledge Area 6.1: Software Design Principles

This knowledge area describes software design principles as incorporated in the PSP.

Key Concept Number and Name	Description

6.1.1 Definition of software design	A *software design* transforms an ill-defined requirement into an implementable product specification.
6.1.2 The role of design in the overall software development process	Software design links the requirements for a system to its implementation. By appropriate use of abstraction, it manages complexity and ensures that the system components work together to produce the desired results.
6.1.3 The "requirements uncertainty principle"	Because a new system affects the users and changes their needs, the requirements for a software system are often not completely known until after the completed product is put to use. The design process must provide a stable basis for this ongoing evolution.
6.1.4 The role of design in PSP	Well-designed components are critical to the success of the larger systems that utilize them. Individual software engineers must employ design practices that can meet the demands of complex and dynamically evolving systems.
6.1.5 Design methodology in PSP	The PSP is independent of design methodology, but does define what design information must be documented.
6.1.6 Design specification structure	The elements of a complete design can be specified using the following specification structure. • external-static (inheritance, class structure) • external-dynamic (services, messages) • internal-static (attributes, program structure, logic) • internal-dynamic (state machine)
6.1.7 Need for design precision	A design specification should be precise. The lack of a precise design is the source of many implementation errors. For best design precision, specify and document design decisions before beginning the coding step of the process.

Knowledge Area 6.2: Design Strategies

Design is a complex intellectual process that cannot be reduced to a mechanical procedure, but some guidelines and strategies can be helpful. This knowledge area addresses the design strategies used in the PSP.

Key Concept Number and Name	Description
6.2.1 Nature of the design process	Design is a learning process that commonly requires moving between design levels and from one part of the system to another.
6.2.2 Design process guidelines	• Where practical, complete higher-level designs first. • Record all assumptions, outstanding issues, questions, and problems. • Where appropriate, use prototyping or experimentation to reduce uncertainty before designing. • Do not consider a design complete until the designs for all interdependent components are also completed. • Document all designs (see Knowledge Area 6.6).

Key Concept Number and Name	Description
6.2.3 Example design strategies	• progressive • functional enhancement • fast-path • dummy (pseudocode)

Key Skill Number and Name	Description
6.2.4 Define a design strategy	Define a design strategy for a specific software development project and justify its use.

Knowledge Area 6.3: Design Quality

This knowledge area describes the key characteristics that can be used to assess the quality of a software design.

Key Concept Number and Name	Description
6.3.1 Design precision	Designs should be concise and unambiguous. The design should contain sufficient detail for all intended uses of the design documentation.
6.3.2 Design completeness	• All relevant details should be included, without any unnecessary redundancy. • The design documentation should not be limited to individual component designs, but should also document system-wide or emergent concerns. • It is helpful to include the rationale for design decisions; it is often helpful to document alternatives that were not chosen.
6.3.3 Design usability	The design must be accessible to and understandable by all its users.

Key Skill Number and Name	Description
6.3.4 Design review	Review the design for precision, completeness, and usability (see Knowledge Area 5.5).

Knowledge Area 6.4: Design Documentation

Software designs must be documented, along with the related requirements, constraints, and rationale. This knowledge area discusses the design documentation that is the responsibility of the individual software engineer.

Key Concept Number and Name	Description
6.4.1 Overall design documentation concerns	Design documentation must be self-consistent, and changes must be managed.
6.4.2 Common types of design documentation	The engineer produces design documentation covering • program context • program structure • related components • external variables, calls, references • detailed program logic description
6.4.3 Design visibility	The design documentation provides the visible representation of a design used for review and verification. The design is recorded using an appropriate design notation (see Key Concept 6.5.1).

Knowledge Area 6.5: Design Templates

The PSP does not specify design techniques, but does provide a set of templates as a frame for design documentation. The templates help to ensure that a system and its modules are completely and precisely implemented.

Key Concept Number and Name	Description
6.5.1 Design notation	A notation based on mathematical logic can assist in producing concise and unambiguous design documentation. The following criteria should be followed when selecting an appropriate design notation. • The design notation should be capable of precisely and completely representing the design. • It must be understandable and usable to the people who will use and/or implement the design. • It should help the designer to produce a high-quality design. • It should be compatible with the implementation language that will be used. • It should allow variable degrees of implementation dependence.
6.5.2 PSP design templates	The PSP design templates represent the static structure and the dynamic behavior of a software system, capturing both the externally visible characteristics and the internal details (see Key Concept 6.1.6). A complete PSP design should contain the following four categories of design elements. • *external-dynamic*: Use the operational specification template (OST) and the functional specification template (FST) to record this information (see Key Concepts 6.5.3 and 6.5.4). • *external-static*: Use the functional specification template (FST) to record this information (see Key Concept 6.5.4). • *internal-dynamic*: Use the state specification template (SST) to record this information (see Key Concept 6.5.5). • *internal-static*: Use the logic specification template (LST) to record this information (see Key Concept 6.5.6).

Key Concept Number and Name	Description
6.5.3 Operational specification template (OST)	The OST documents the external-dynamic characteristics of a part of a software system. It describes one or more scenarios involving the part and the actors (e.g., users or other systems) that interact with it. Each OST has a unique ID, a user objective, and a scenario objective. For each step in a scenario, the OST lists • source (system or specified actor) • step number • action • comments
6.5.4 Functional specification template (FST)	The FST documents a part (e.g., a class) of a software system, its external-static relationships, and its externally visible attributes. The FST also documents the external-dynamic characteristics of a part. It describes actions (e.g., class methods) that the part makes available for external use; this description includes the defined interface for each action, including arguments, constraints, and returned results.
6.5.5 State specification template (SST)	The SST documents the internal-dynamic behavior of a software system and its parts (e.g., classes) when that behavior is represented as a set of states, transitions between states, and actions associated with the transitions. The SST can be supplemented by a separate state diagram that graphically depicts the states, transition conditions, and actions. An SST contains • state names and descriptions • functions and internal parameters used in transition conditions • details of state transitions – current state – next state – transition condition (predicate) – action performed when transition occurs
6.5.6 Logic specification template (LST)	The LST documents the internal-static characteristics of a part of a software system. It describes the internal logic of the part, using pseudocode to clearly and concisely explain its operation. Note that the LST information may be embedded as comments in the program source code, rather than using a separate form, as long as it is clear and sufficiently detailed.
6.5.7 Template usage	The PSP design templates may be • used to document a design produced using various design techniques • used to document the design of an existing software system, to support redesign or verification • supplemented or partially replaced by other design documentation techniques (e.g., the Unified Modeling Language), as long as equivalent design information is captured in an easily usable form • applied at different levels of the design hierarchy

Key Skill Number and Name	Description
6.5.8 Select a design notation	Use the guidelines outlined in Key Concept 6.5.1 to choose a design notation that is appropriate to the design requirements, implementation language, and other parameters as specified.
6.5.9 Create an OST	1. Determine the scenarios that describe the external-dynamic characteristics of a module. 2. Identify the purpose of each scenario from the point of view of the user or other actor. 3. Describe the design purpose for creating this OST. 4. For each action step, document the action source, step number, and action, with explanatory comments as appropriate.
6.5.10 Create an FST	1. Determine the external-static and external-dynamic characteristics of a module. 2. List its externally visible attributes, with declarations and descriptions. 3. Document externally available actions, with names, arguments, return values, and constraints. Use a precise design notation wherever possible.
6.5.11 Create an SST	1. Determine the internal-dynamic characteristics of a module. 2. Name and describe its states. 3. List functions and internal parameters used in transition conditions. 4. Document all transitions from all states, with current state, next state, transition condition, and associated action. Note clearly any impossible or forbidden transitions.
6.5.12 Create an LST	1. Determine the internal-static characteristics of a module. 2. Reference other related specifications. 3. Describe internal parameters. 4. Using pseudocode, clearly explain what the program is to do and how.

Knowledge Area 6.6: Design Verification

To be effective, design reviews must go beyond simply reading through a design document. The PSP provides a number of design verification techniques that can be used to identify errors and omissions in software designs.

Key Concept Number and Name	Description
6.6.1 Design standards	Software designs can be verified against standards, which promote consistency and quality. These standards may include • product conventions • product design standards • reuse standards

Key Concept Number and Name	Description
6.6.2 Verification methods	• execution table verification • trace-table verification • state-machine verification • loop verification • other analytical verification methods

Key Skill Number and Name	Description
6.6.3 Choose the appropriate design verification method	1. Analyze your personal defect data to determine which design aspects are most defect-prone. It is not a prudent use of time to verify design aspects where you make few (if any) defects. 2. Evaluate the effectiveness of current verification methods. Identify a family of effective techniques and use them, even on small programs. 3. Consider the economics of current verification techniques. Choose the verification methods that are most effective personally and that best apply to the conditions of the design.
6.6.4 Use execution table verification	1. Identify loops and complex routines for verification. 2. Choose order of analysis (e.g., top down or bottom up). 3. Construct an execution table with program steps and relevant variable values, using multiple copies for loop iterations. 4. Verify execution results against the requirements specification.
6.6.5 Use trace-table verification	1. Identify representative logical cases for analysis. 2. For each logical case, verify using an execution table.
6.6.6 Execution table verification vs. trace-table verification	Differentiate between execution table and trace-table verification and know when to use each one.
6.6.7 Use state-machine verification	1. Check the state-machine structure to ensure it has no hidden traps or loops, using a state diagram if practical. 2. Examine each state and verify that the set of transitions from that state is complete (defined for all possible transition condition values). 3. Examine each state and verify that the associated state transitions are orthogonal (only one transition defined for each set of transition condition values).
6.6.8 Use loop verification	Verify loop initiation, incrementing, and termination, using the verification methods appropriate to the type of loop. • for-loop verification • while-loop verification • repeat-until verification

Competency Area 7: Process Extensions and Customization

Competency Area Number and Name	7. Process Extensions and Customization
Competency Area Description	This competency area describes the modifications to the PSP that are required when scaling up from smaller programs to larger ones, when working with unfamiliar situations or environments, or when moving to team-based development instead of working alone.
Knowledge Areas	7.1 Defining a Customized Personal Process 7.2 Process Evolution 7.3 Advanced Process Applications 7.4 Professional Responsibility
References	[Horn 90] [Humphrey 95, pp. 483-485, 725-740] [Humphrey 05a, Chapter 13]

DESCRIPTIONS OF THE PROCESS EXTENSIONS AND CUSTOMIZATION KNOWLEDGE AREAS

Knowledge Area Number and Name	Description
7.1 Defining a Customized Personal Process	A defined process should not be regarded as "one size fits all." This knowledge area addresses situations in which processes must be customized to meet changes in needed outputs or developed from the ground up to address new situations or environments.
7.2 Process Evolution	A process cannot be evolved to fit changing needs or situations until the current process accurately represents what is actually done when using that process. This knowledge area addresses the activities involved with incrementally evolving an initial process into one that is an accurate and complete description of the actual process.
7.3 Advanced Process Applications	Experienced PSP users may encounter situations when the original PSP processes may not be conveniently applicable for planning and developing a product. Modifications of the PSP, such as the Prototype Experimental Process (PEP) and the Product Maintenance Process (PMP), allow the application of PSP concepts and skills to such situations.
7.4 Professional Responsibility	Exceptional software development work requires responsible behavior on the part of the software professional. This knowledge area describes some of the practices of responsible software professionals.

Knowledge Area 7.1: Defining a Customized Personal Process

A defined process should not be regarded as "one size fits all." This knowledge area addresses situations in which processes must be customized to meet changes in needed outputs or are developed from the ground up to address new situations or environments.

Key Concept Number and Name	Description

Key Concept Number and Name	Description
7.1.1 When to define a customized process	Different programming situations call for different programming methods: what works well in one environment may not be effective in another. For example, simple programming tasks may require little or no design time. However, larger systems or high-security systems (regardless of size), require a thorough design. A process without a design phase may require customization to include this activity when tailoring an existing process to fit a new situation, when the process scalability changes, or when security requirements change.
7.1.2 Scalability	*Scalability* is the scope or breadth of activities for which a process is designed. Changes in scalability may include changing the ranges in numbers of people, product size, product schedule, project life cycle, or development environment for which the process is fit and precise.
7.1.3 Tailoring	*Tailoring* is the act of adapting process designs and process definitions to support the enactment of a process for a particular purpose.
7.1.4 Process customization steps	Software process definition is much like software development: start with the users' needs and end with final test and release. There are eight general steps for personal process development. 1. Determine your needs and priorities. 2. Define process objectives, goals, and quality criteria. 3. Characterize your current process. 4. Characterize your target process. 5. Establish a process development strategy. 6. Define your initial process. 7. Validate your initial process. 8. Enhance your process.
7.1.5 Information mapping principles and defining a customized process	When defining a customized process or developing scripts and forms from scratch, it is wise to follow the following principles of information mapping [Horn 90]. • *Chunking*: Organize information into groups that are manageable to read and/or to accomplish. • *Relevance*: Group "like things" together and exclude unrelated items from each chunk. • *Labeling*: Provide the user with a label for each chunk of information. • *Consistency*: Use consistent terms within each chunk of information, between the chunk and the label, in organizing the information, and in formatting the document or instrument in which the information is recorded. • *Integrate graphics*: Use tables, illustrations, and diagrams as an integral part of the writing. • *Accessible detail*: Write at the level of detail that makes the document usable for all readers. • *Hierarchy of chunking and labeling*: Group small chunks around a single relevant topic and provide each group with a label.

Key Skill Number and Name	Description

7.1.6 Define a customized process	Given a set of user requirements, modify an existing process or define a new process to be used in producing the required outcomes.

Knowledge Area 7.2: Process Evolution

A process cannot be evolved to fit changing needs or situations until the current process accurately represents what is actually done when using that process. This knowledge area addresses the activities involved with incrementally evolving an initial process into one that is an accurate and complete description of the actual process.

Key Concept Number and Name	Description
7.2.1 Initial process definition	Initial process descriptions are seldom accurate, due to a phenomenon analogous to the Heisenberg Uncertainty Principle: the act of defining a process changes that process. The initial description of the process usually contains omissions, idealizations, and other inaccuracies. The process of accurately describing what really happens often affects the process during the very act of defining it.
7.2.2 Developing an actual or "matured" personal process	1. Start with a characterization of the process as currently used. 2. Define the target or ideal process. 3. Define the steps needed to move from the current process to the target process. 4. Develop the necessary scripts, forms, standards, and measures to use in your process. 5. Measure your progress towards the ideal. Gather data as you use the current process and identify areas for changes. Ask questions (e.g., What steps are ill defined? What steps give the most trouble? Where do most defects arise?) to identify areas that need to be improved or modified. 6. Focus on the areas that give the most trouble. Implement refinements a little at a time and adjust the process as needed, based on your data. 7. Continue to implement changes and refinements a few at a time until the final representation is close to or at your target process.
7.2.3 Evolving the "matured" process	See Knowledge Areas 2.3 and 2.4.

Key Skill Number and Name	Description
7.2.4 Define an initial process	Be able to define a process as used in everyday practice. Develop scripts, forms, standards, and measures as necessary to support the process.
7.2.5 Evolve the initial process	See Knowledge Areas 2.3 and 2.4.

Knowledge Area 7.3: Advanced Process Applications

Experienced PSP users may encounter situations when the original PSP processes may not be conveniently applicable for planning and developing a product. Modifications of the PSP,

such as the Prototype Experimental Process (PEP) and the Product Maintenance Process (PMP), allow the application of PSP concepts and skills to such situations.

Key Concept Number and Name	Description
7.3.1 Prototype Experimental Process (PEP)	Use the PEP when working in unfamiliar programming environments or when building prototype systems with loosely defined or poorly understood requirements.
7.3.2 Product Maintenance Process (PMP)	Use the PMP when making modifications, enhancements, or repairs to legacy systems with large or defective base code.

Knowledge Area 7.4: Professional Responsibility

Exceptional software development work requires responsible behavior on the part of the software professional. This knowledge area describes some of the practices of responsible software professionals.

Key Concept Number and Name	Description
7.4.1 Use effective methods in your work	Good software practices are straightforward, but few people consistently use them. The dedicated professional finds effective methods for consistently producing high-quality software and then uses them.
7.4.2 Use data to discover your strengths and weaknesses	Use the postmortem analysis of your personal data to build an understanding of what you do well and areas where you need to improve. Focus on regularly making small improvements and major changes will take care of themselves.
7.4.3 Practice	The key to improving your software development products is to practice your skills on the job to the maximum extent possible.
7.4.4 Learn from others, and pass on what you know	Talk to your colleagues and review the literature to learn about new techniques and to learn from the mistakes of others. As you learn and build your own knowledge, share what you have learned with others. Take advantage of what you find and contribute what you learn.
7.4.5 Find and learn new methods	Watch for innovations that are pertinent to your personal needs. Allocate time in your schedule for skill building whenever possible. By keeping up to date, you make yourself more attractive to your current employer (and to future employers) as a desirable and competent professional.

5 Conclusion

The PSP BOK is a three-tiered hierarchical model delineating the competencies, knowledge areas, and key concepts and skills that compose the essential proficiencies to be mastered by a PSP-trained software engineer. The content is drawn from the work of Watts S. Humphrey over the past decade. As PSP adoption continues to grow, it is expected that the PSP BOK will evolve to reflect further process extensions and a deeper understanding of the application of the key concepts and skills in actual on-the-job practice.

Appendix Key Statistical Formulae and Procedures

PSP emphasizes use of statistical methods in implementing and improving personal software engineering processes. Many of the specific formulae and statistical procedures delineated below are embodied within PSP support tools. The specific statistical formulae and procedures are included in this body of knowledge because they are important concepts and skills for PSP instructors and tool developers. Another reason for including the formulae is that many PSP courses and/or curricula include a requirement for learners to develop programs to implement these statistical formulae and procedures. This provides the students with data to help them understand and improve their personal processes, and gives them a deeper understanding of the mechanics behind the PSP methods and procedures. PSP engineers are not expected to be able to recall the information in this section, but they are expected to be able to use these methods appropriately.

Numerical data for the PSP are assumed to be generated by some common process, so they can be considered to come from a distribution. Thus, statistical formulae can enable inference about the underlying distribution and the process that generated it. Therefore, the user is dealing with estimates of the distribution parameters. Although not stated for each formula, the statistics are estimates of the various statistical parameters, not their actual values.

Key Statistical Formula and Procedure	Description and Formula
A.1 Calculate the mean, μ, of a dataset $x_1,...,x_n$	$\mu = x_{avg} = \frac{1}{n}\sum_{i=1}^{n} x_i$
A.2 Calculate the variance, σ^2, about the mean for the dataset in A.1	$\sigma^2 = \frac{1}{n-1}\sum_{i-1}^{n}(x_i-\mu)^2$
A.3 Calculate the standard deviation, σ, for the variance in A.2	$\sigma = \sqrt{\sigma^2}$
A.4 Transform a dataset $x_1,...,x_n$ with mean μ and standard deviation σ into standard normal form, *i.e.*, mean = 0 and standard deviation = 1.s	$z_i = \frac{x_i - \mu}{\sigma}$
A.5 Calculate the correlation between data pairs $(x_1, y_1), \ldots, (x_n, y_n)$	$r_{x,y} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$ A value of r^2 greater than or equal to 0.5 implies a high positive linear relationship. Although r^2 may approach 1, the relationship may not be significant (see A.8).

Key Statistical Formula and Procedure	Description and Formula
A.6 The gamma function	$\Gamma(x) = (x-1)\Gamma(x-1)$, where For integer values of x, $\Gamma(x) = (x-1)!$ Also, $\Gamma(1) = 1$ $\Gamma(1/2) = \sqrt{\pi}$
A.7 The t-function (probability density) for degrees of freedom *df*	$F(x) = \frac{\Gamma\left(\frac{df+1}{2}\right)}{\sqrt{df * \pi}\,\Gamma\left(\frac{df}{2}\right)}\left(1+\frac{x^2}{df}\right)^{-\left(\frac{df+1}{2}\right)}$
A.8 Calculate the significance of the correlation in A.5	$t = \frac{\lvert r_{x,y}\rvert\sqrt{df}}{\sqrt{1-r_{x,y}^{\ 2}}}, df = n-2$ 1. Calculate 2. Calculate the probability *p* of the value of *t* in step 1 by integrating the *t* density function in A.7 from -*t* to *t* for *df* = *n–2*. $p = \int_{-t}^{t} F(x)dx$ 3. Calculate the area under the two tails of the *t* function. *p* is the area under the *t* function from -*t* to *t*. The area under the tails of the *t* function is the sum of the integrals from -∞ to -*t* and *t* to ∞. Because the total probability (area from -∞ to ∞) is 1.0, the area under the two tails is 1.0 – p(*t*). 4. The correlation in A.5 is significant if 1 – p < 0.05
A.9 Calculate linear regression estimating parameters for data pairs $(x_1, y_1), \ldots, (x_n, y_n)$	Calculate the linear regression estimating parameters β_0 and β_1. $\beta_1 = \frac{\sum_{i=1}^{n} x_i y_i - n x_{avg} y_{avg}}{\sum_{i=1}^{n} x_i^2 - n x_{avg}^2}$ $\beta_0 = y_{avg} - \beta_1 x_{avg}$
A.10 Use linear regression to calculate an estimated value for *y* given an estimated value of *x* and estimating parameters from A.9	$y = \beta_0 + \beta_1 x$

Key Statistical Formula and Procedure	Description and Formula
A.11 Calculate multiple regression estimating parameters β_0, β_1, …, β_n for data set (x_{11}, …, x_{m1}, y_1), (x_{12}, …, x_{m2}, y_2), …, (x_{1n}, …, x_{mn}, y_n)	Calculate the multiple-regression estimating parameters β_0, β_1, … , β_n. Find the beta parameters by solving the following simultaneous linear equations. $\beta_0 n + \beta_1 \sum_{i=1}^{n} x_{1i} + \beta_2 \sum_{i=1}^{n} x_{2i} + \ldots + \beta_m \sum_{i=1}^{n} x_{mi} = \sum_{i=1}^{n} y_i$ $\beta_0 \sum_{i=1}^{n} x_{1i} + \beta_1 \sum_{i=1}^{n} x_{1i}^2 + \beta_2 \sum_{i=1}^{n} x_{1i}x_{2i} + \ldots + \beta_m \sum_{i=1}^{n} x_{1i}x_{mi} = \sum_{i=1}^{n} x_{1i}y_i$ $\beta_0 \sum_{i=1}^{n} x_{2i} + \beta_1 \sum_{i=1}^{n} x_{2i}x_{1i} + \beta_2 \sum_{i=1}^{n} x_{2i}^2 + \ldots + \beta_m \sum_{i=1}^{n} x_{2i}x_{mi} = \sum_{i=1}^{n} x_{2i}y_i$ … $\beta_0 \sum_{i=1}^{n} x_{mi} + \beta_1 \sum_{i=1}^{n} x_{mi}x_{m1} + \beta_2 \sum_{i=1}^{n} x_{mi}x_{m2} + \ldots + \beta_m \sum_{i=1}^{n} x_{mi}^2 = \sum_{i=1}^{n} x_{mi}y_i$
A.12 Use multiple regression to calculate an estimated value for y given an estimated value of (x_1, …, x_m) and estimating parameters β_0, …, β_m from A.11	$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_m x_m$
A.13 Calculate the standard normal function.	$F(x) = \frac{1}{\sqrt{(2\pi)}} e^{-\left(\frac{x^2}{2}\right)}$
A.14 Calculate the chi-square function for *df* degrees of freedom	$F(x) = \frac{1}{2^{\left(\frac{df}{2}\right)} \Gamma\left(\frac{df}{2}\right)} x^{\left(\frac{df}{2}\right)-1} e^{-\left(\frac{x}{2}\right)}$
A.15 Approximate the integral of a function *F(x)* from *a* to *b* using Simpson's Rule	$\int_a^b F(x)dx \approx \frac{W}{3}\left[F(a) + \sum_{i=1,3,5\ldots}^{N-1} 4F(iW) + \sum_{i=2,4,6\ldots}^{N-2} 2F(iW) + F(b)\right]$ where N is the number of segments (an even number) W = $\frac{b-a}{N}$

Key Statistical Formula and Procedure	Description and Formula
A.16 Calculate the range around the new estimate $\bar{x}$ for the 70% prediction interval for linear regression applied to the data pairs $(x_1, y_1), \ldots, (x_n, y_n)$.	$$Range = t(0.70, df)\sigma\sqrt{1+\frac{1}{n}+\frac{\left(\bar{x}-x_{avg}\right)^2}{\sum_{i=1}^{n}\left(x_i-x_{avg}\right)^2}}$$ where 1. t is the limit of the integration of the t density function for $df = n\text{-}2$ (see A.7) such that $$\int_{-t}^{t} F(x)dx = 0.70.$$ 2. $$\sigma^2 = \left(\frac{1}{df}\right)\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$$ is the variance of y_i from the regression line for the data.
A.17 Calculate the range around the new estimate $(\bar{x}_1, \ldots, \bar{x}_m)$ for the 70% prediction interval for multiple regression applied to the data set $(x_{11}, \ldots, x_{m1}, y_1)$, $(x_{12}, \ldots, x_{m2}, y_2), \ldots, (x_{1n}, \ldots, x_{mn}, y_n)$	$$Range = t(0.70, df)\sigma\sqrt{1+\frac{1}{n}+\frac{\left(\bar{x}_1-x_{1,avg}\right)^2}{\sum_{i=1}^{n}\left(x_{1i}-x_{1,avg}\right)^2}+\ldots+\frac{\left(\bar{x}_m-x_{m,avg}\right)^2}{\sum_{i=1}^{n}\left(x_{mi}-x_{m,avg}\right)^2}}$$ where 1. $x_{k,avg} = \frac{1}{n}\sum_{i=1}^{n} x_{ki}$, $k = 1, \ldots, m$ 2. t is the limit of the integration of the t density function for $df = n\text{-}m\text{-}1$ (see A.7) such that $$\int_{-t}^{t} F(x)dx = 0.70.$$ 3. $$\sigma^2 = \left(\frac{1}{df}\right)\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x_{1i} - \ldots - \beta_m x_{mi}\right)^2$$ is the variance of y_i from the regression line for the data.

Key Statistical Formula and Procedure	Description and Formula
A.18 Use the chi-square test for normality of *n* data points	1. Convert data to standard normal form. 2. Divide the normal distribution into some number of segments *S*, such that • each segment has probability 1/*S* • *n/S* ≥ 5 (an integer if possible) • *S* > 3 • $S^2 \geq n$ • when more than one value of *S* is possible, select the one such that *n* and S^2 are most nearly equal 3. Determine how many items of the ideal normal distribution would fall into each segment. This number is N_i. (If *n/S* is an integer, then $N_i = n/S$). 4. Using the same segment boundaries, determine how many items from the normalized data set fall into each segment. This number is k_i. 5. Calculate the *Q* value for the segments. $Q = \sum_{i=1}^{S} \frac{(N_i - k_i)^2}{N_i}$ 6. Calculate the probability *p* of the χ^2 distribution for S-1 degrees of freedom (*df*) by integrating the chi-square function (see A.14) from 0 to *Q*. 7. Calculate the distribution tail as *1-p*. 8. Tail areas greater than 0.2 are generally considered sufficient to accept a fit; tail areas less than 0.05 are generally considered sufficient to reject a fit; intermediate values indicate intermediate degrees of fit.

Bibliography

URLs are valid as of the publication date of this document.

[Davis 03] Davis, Noopur & Mullaney, Julia. *The Team Software Process (TSP) in Practice: A Summary of Recent Results* (CMU/SEI-2003-TR-014, ADA418430). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr014.html

[Feiler 92] Feiler, Peter H. & Humphrey, Watts S. *Software Process Development and Enactment: Concepts and Definitions* (CMU/SEI-92-04). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.004.html

[Ford 96] Ford, Gary & Gibbs, Norman E. *A Mature Profession of Software Engineering* (CMU/SEI-96-TR-004, ADA307889). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.004.html

[Hayes 97] Hayes, Will & Over, James. *The Personal Software Process: An Empirical Study of the Impacts of PSP on Individual Engineers* (CMU/SEI-97-TR-001, ADA335543). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. http://www.sei.cmu.edu/publications/documents/97.reports/97tr001/97tr001abstract.html

[Hilburn 99] Hilburn, Thomas B.; Hirmanpour, Iraj; Khajenoori, Soheil; Turner, Richard; & Qasem, Abir. *A Software Engineering Body of Knowledge Version 1.0* (CMU/SEI-99-TR-004, ADA363793). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. http://www.sei.cmu.edu/publications/documents/99.reports/99tr004/99tr004abstract.html

[Horn 90] Horn, Robert E. *Developing Procedures, Policies, and Documentation.* Waltham, MA: Information Mapping, Inc., 1990.

[Humphrey 89] Humphrey, Watts S. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.

[Humphrey 95] Humphrey, Watts S. *A Discipline for Software Engineering*. Reading, MA: Addison-Wesley, 1995.

[Humphrey 97] Humphrey, Watts S. *Introduction to the Personal Process*. Reading, MA: Addison-Wesley, 1997.

[Humphrey 00] Humphrey, Watts S. *The Personal Software Process (PSP)* (CMU/SEI-2000-TR-022, ADA387268). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/publications/documents/00.reports/00tr022.html

[Humphrey 05a] Humphrey, Watts S. *PSP: A Self-Improvement Process for Software Engineers*. Reading, MA: Addison-Wesley, 2005.

[Humphrey 05b] Humphrey, Watts S. *PSP: A Self-Improvement Process for Software Engineers – Instructor's Guide*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[IEEE 04] IEEE Computer Society. *Guide to the Software Engineering Body of Knowledge (SWEBOK) 2004 Version*. http://www.swebok.org/home.html (2004).

[Naur 69] Naur, Peter & Randell, Brian, eds. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee*. Garmisch, Germany, Oct. 7-11, 1968. Brussels, Belgium: Scientific Affairs Division, NATO, 1969.

[PMI 04] Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, Third Edition. Newton Square, PA: PMI Publishing Division, 2004.

[Webb 99] Webb, David & Humphrey, Watts S. "Using the TSP on the Task-View Project." *CrossTalk 12*, 2 (February 1999): 3-10.

[Wikipedia 05] Wikipedia, The Free Encyclopedia. *Criticism of software engineering*. http://en.wikipedia.org/wiki/Criticism_of_software_engineering (2005).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	February 2008 (updated March 2008)	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
The Personal Software Process℠ (PSP℠) Body of Knowledge, Version 1.0	F19628-00-C-0003

6. AUTHOR(S)

Marsha Pomeroy-Huff, Julia Mullaney, Robert Cannon, Mark Seburn

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2005-SR-003

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

As the profession of software engineering evolves and matures, it must achieve some of the critical elements needed for recognition as a bona fide discipline. Among these elements are the establishment of a recognized body of knowledge (BOK) and certification of professional practitioners.

The body of knowledge contained in this report is designed to complement the IEEE Computer Society's *Software Engineering Body of Knowledge (SWEBOK)* by delineating the key skills and concepts that compose the knowledge areas and competencies of a proven-effective process improvement method, the Personal Software Process (PSP). As adoption of the PSP methodology continues to grow, it becomes crucial to document the fundamental knowledge and skills that set PSP practitioners apart from other software engineers. The PSP BOK serves this purpose and more. It helps individual practitioners to assess and improve their own skills; provides employers with an objective baseline for assessing the personal process skills and capabilities of their engineers and product development teams; and guides academic institutions that want to incorporate PSP into their software and other engineering courses or curricula. The PSP BOK also facilitates the development of PSP certification programs that are based on a well-established, standard set of knowledge and skills.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Personal Software Process, PSP, quality, quality software, software development, software engineer, software engineering, software process, data collection, process quality, development, software process improvement, best practices	90

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL